

Android应用开发教程

◎ 赵明渊 主编





清华大学出版社

21 世纪高等学校计算机专业实用规划教材

Android 应用开发教程

赵明渊　主编

清华大学出版社
北　京

内 容 简 介

本书基于 Android Studio 和 Eclipse 开发环境，介绍 Android 系统体系结构和应用开发环境，Android 应用的创建、调试和发布，Activity、Fragment 和 Intent，Android 基本控件、高级控件，Android 事件处理，后台服务，数据存储，多媒体服务，定位服务，应用项目开发等内容。

本书注重理论与实践的结合，采取“项目驱动”的方式进行讲述。本书在作者多年教学和软件开发经验的基础上，讲解详细深入，论述通俗易懂，具备编程基础的读者，通过本书的学习都可掌握 Android 软件开发。为方便教学，每章都有大量示范性设计实例和运行结果，所有实例都经过调试通过，书后还附有习题答案。

本书可作为大学本科、高职、高专及培训班课程的教学用书，也适于计算机应用人员和计算机爱好者自学参考。

本书提供的教学课件、所有实例的源代码的下载网址为 http://www.tup.com.cn。

图书在版编目(CIP)数据

Android 应用开发教程/赵明渊主编. —北京：清华大学出版社，2018
(21 世纪高等学校计算机专业实用规划教材)
ISBN 978-7-302-48318-2

Ⅰ. ①A…　Ⅱ. ①赵…　Ⅲ. ①移动终端—应用程序—程序设计—教材　Ⅳ. ①TN929.53

中国版本图书馆 CIP 数据核字(2017)第 218500 号

责任编辑：魏江江　薛　阳
封面设计：刘　键
责任校对：时翠兰
责任印制：李红英

出版发行：清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦 A 座　　**邮　　编：**100084
社 总 机：010-62770175　　**邮　　购：**010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质量反馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
课件下载：http://www.tup.com.cn，010-62795954
印 装 者：清华大学印刷厂
经　　销：全国新华书店
开　　本：185mm×260mm　　**印　　张：**29.5　　**字　　数：**718 千字
版　　次：2018 年 5 月第 1 版　　**印　　次：**2018 年 5 月第1 次印刷
印　　数：1～1500
定　　价：79.50 元

产品编号：068612-01

出版说明

随着我国改革开放的进一步深化，高等教育也得到了快速发展，各地高校紧密结合地方经济建设发展需要，科学运用市场调节机制，加大了使用信息科学等现代科学技术提升、改造传统学科专业的投入力度，通过教育改革合理调整和配置了教育资源，优化了传统学科专业，积极为地方经济建设输送人才，为我国经济社会的快速、健康和可持续发展以及高等教育自身的改革发展做出了巨大贡献。但是，高等教育质量还需要进一步提高以适应经济社会发展的需要，不少高校的专业设置和结构不尽合理，教师队伍整体素质亟待提高，人才培养模式、教学内容和方法需要进一步转变，学生的实践能力和创新精神亟待加强。

教育部一直十分重视高等教育质量工作。2007 年 1 月，教育部下发了《关于实施高等学校本科教学质量与教学改革工程的意见》，计划实施“高等学校本科教学质量与教学改革工程(简称‘质量工程’)”，通过专业结构调整、课程教材建设、实践教学改革、教学团队建设等多项内容，进一步深化高等学校教学改革，提高人才培养的能力和水平，更好地满足经济社会发展对高素质人才的需要。在贯彻和落实教育部“质量工程”的过程中，各地高校发挥师资力量强、办学经验丰富、教学资源充裕等优势，对其特色专业及特色课程(群)加以规划、整理和总结，更新教学内容，改革课程体系，建设了一大批内容新、体系新、方法新、手段新的特色课程。在此基础上，经教育部相关教学指导委员会专家的指导和建议，清华大学出版社在多个领域精选各高校的特色课程，分别规划出版系列教材，以配合“质量工程”的实施，满足各高校教学质量和教学改革的需要。

本系列教材立足于计算机专业课程领域，以专业基础课为主、专业课为辅，横向满足高校多层次教学的需要。在规划过程中体现了如下一些基本原则和特点。

(1) 反映计算机学科的最新发展，总结近年来计算机专业教学的最新成果。内容先进，充分吸收国外先进成果和理念。

(2) 反映教学需要，促进教学发展。教材要适应多样化的教学需要，正确把握教学内容和课程体系的改革方向，融合先进的教学思想、方法和手段，体现科学性、先进性和系统性，强调对学生实践能力的培养，为学生知识、能力、素质协调发展创造条件。

(3) 实施精品战略，突出重点，保证质量。规划教材把重点放在公共基础课和专业基础课的教材建设上；特别注意选择并安排一部分原来基础比较好的优秀教材或讲义修订再版，逐步形成精品教材；提倡并鼓励编写体现教学质量和教学改革成果的教材。

(4) 主张一纲多本，合理配套。专业基础课和专业课教材配套，同一门课程有针对不同层次、面向不同应用的多本具有各自内容特点的教材。处理好教材统一性与多样化，基本教材与辅助教材、教学参考书，文字教材与软件教材的关系，实现教材系列资源配套。

(5) 依靠专家，择优选用。在制定教材规划时要依靠各课程专家在调查研究本课程教材建设现状的基础上提出规划选题。在落实主编人选时，要引入竞争机制，通过申报、评审确定主题。书稿完成后要认真实行审稿程序，确保出书质量。

繁荣教材出版事业，提高教材质量的关键是教师。建立一支高水平教材编写梯队才能保证教材的编写质量和建设力度，希望有志于教材建设的教师能够加入到我们的编写队伍中来。

21 世纪高等学校计算机专业实用规划教材

联系人：魏江江 weijj@tup.tsinghua.edu.cn

前　言

跨入移动互联网时代，智能手机、平板电脑等智能移动设备走入了千家万户，随之而来的是移动平台下的应用软件开发需求日益旺盛，移动互联网与 Android 带来更多的就业机会与创业机会。Android 应用开发是 IT 企业招聘人才的主要测试内容，我国许多高校包括计算机专业、通信专业、电子商务专业、电子信息专业、软件工程专业及其相关专业纷纷开设 Android 课程。

Android 是当前的主流移动操作系统，又是强大的手机应用开发平台。它建立在 Java 基础之上，提供了应用开发框架、应用开发和调试工具，成为一个新兴的热点和软件行业的一股新兴力量。

本书全面系统地介绍 Android 应用开发，将基础知识和实际应用有机结合起来，以利于培养读者的理解能力和应用系统开发能力。本书基于 Android Studio 和 Eclipse 开发环境，深入浅出地介绍 Android 应用开发的各个要点，具体介绍 Android 系统体系结构和应用开发环境，Android 应用的创建、调试和发布，Activity、Fragment 和 Intent，Android 基本控件，Android 事件处理、高级控件和菜单，后台服务，数据存储，多媒体服务，定位服务和百度地图应用开发，Android 应用项目开发等内容。

本书特色如下：

- 解题思路清晰，程序分析详细。在每一个实例中，注重清晰的解题思路，并详细讲解开发步骤和进行程序分析。
- 方便教学，资源配套。本教程免费提供教学课件、所有实例的源代码，章末习题有选择题、填空题、问答题和应用题等类型，书后附习题答案，以供教学参考。
- 理论与实践相结合，项目驱动。以利于学生对 Android 应用开发的基本概念、原理、方法和技术有较深刻的理解，掌握 Android 应用开发的基本知识和技术，初步具备开发 Android 应用项目的能力。
- 通过 Android 基本控件和高级控件、后台服务、数据存储、多媒体服务等章节的论述和举例，着重培养学生 Android 界面设计和程序设计的能力。
- 通过 Android 应用项目开发实例的论述和分析，培养学生开发一个简单 Android 应用项目的能力。

本书由赵明渊主编，参加本书编写的有余堃、胡宇、马磊、任健、李华春、唐明伟、周亮宇、赵凯文、李文君、程小菊、邓铠凌、王成均。对于帮助完成基础工作的同志，在此表示感谢！

由于作者水平有限，不当之处，敬请读者批评指正。

编　者

2018 年 1 月

目　　录

第 1 章 Android 系统体系架构和应用开发环境

本章要点

- Android 的应用和特点。
- Android 系统的体系架构包括应用程序、应用程序框架、系统库和运行时、Linux 内核等 4 个层次。
- 搭建 Eclipse 集成开发环境需要 JDK、Eclipse、ADT 和 SDK。
- Android Studio 是 Google 为开发设计人员提供的新的集成开发环境。

Android 既是一个开源的手机操作系统,又是一个强大的应用开发平台。本章介绍 Android 概述、Android 操作系统的体系架构、Eclipse 集成开发环境、Android Studio 集成开发环境等内容。

1.1 Android 概述

Android 是一个开源的手机操作系统,使用 Linux 作为操作系统的内核,实现了进程管理、存储管理、设备管理、文件管理等操作系统的基本功能,同时,Android 提供了开放的 Android SDK(Software Development Kit)软件开发组件,开发人员基于 Android 应用开发框架可以方便、灵活地开发各种移动应用。

1.1.1 Android 简介

Android 最初由 Android 公司开发,公司的创始人为 Andy Rubin,后该公司被 Google 公司收购,Andy Rubin 成为 Google 公司的工程部副总裁,继续负责 Android 项目的开发。

Google 公司在 2007 年 11 月发布 Android 1.0 手机操作系统,同时宣布成立开放手机联盟,该联盟由摩托罗拉、索尼爱立信(Sony Ericsson)、韩国三星电子、韩国 LG 电子、中国移动(China Mobile)、英特尔(Intel)等 34 家手机制造商、电信运营商、软件开发商和芯片制造商组成。

2009 年 5 月,Google 公司发布了 Android 1.5,大幅度改进界面,并提供对蓝牙连接的支持。

2011 年 9 月,Google 公司发布了 Android 4.0,综合 Android 2.3 与 Android 3.0 的优点,支持手机设备与平板电脑。

2014 年 10 月,Google 公司发布了 Android 5.0。

2015 年 5 月,Google 公司发布了 Android 6.0。

2016 年 3 月,Google 公司发布了 Android 7.0。

1.1.2 Android 应用

Android 系统平台可以开发出通信、搜索、新闻、娱乐、商务、家电控制、传感器、可穿戴设备等移动服务应用,以及进行 Android 与物联网结合的应用开发。

中国互联网络信息中心在 2014 年 8 月发表的《中国移动互联网调查研究报告》中指出:截至 2014 年 6 月底,我国手机网民规模为 5.27 亿人,较 2013 年年底增加 2699 万人。在我国手机网民常用手机中,以苹果、三星占比最高,分别为 21.2%和 18.4%,华为、小米和联想在手机网民常用手机市场份额中占比相对较高,分别为 8.6%、8.1%和 7.1%。

中国手机网民各类手机网络应用的使用率(2013.6—2014.6)如表 1.1 所示。移动应用对人们的信息、社交、娱乐和购物等各方面产生重要影响。其中,电子商务类应用和娱乐类应用尤为突出,并通过手机打车、手机地图和手机支付等应用加大对社会生活服务的渗透。

表 1.1 2013.6—2014.6 中国手机网民各类手机网络应用的使用率

	2014 年 6 月		2013 年 6 月		
应用	用户规模/万	网民使用率/%	用户规模/万	网民使用率/%	年增长率/%
手机即时通信	45 921	87.1	39 735	85.7	15.6
手机搜索	40 583	77.0	32 431	69.9	25.1
手机网络新闻	39 087	74.2	31 356	67.6	24.7
手机网络音乐	35 462	67.3	24 388	52.6	45.4
手机网络视频	29 378	55.7	15 961	34.4	84.1
手机网络游戏	25 182	47.8	16 128	34.8	56.0
手机网络文学	22 211	42.1	20 370	43.9	9.0
手机网上支付	20 509	38.9	7911	17.1	159.2
手机网络购物	20 499	38.9	7636	16.5	168.5
手机微博	18 851	35.8	22 951	49.5	−17.9
手机网上银行	18 316	34.8	7236	15.6	153.1
手机邮件	14 827	28.1	12 641	27.3	17.3
手机社交网站	13 387	25.4	19 565	42.2	−31.6
手机团购	10 220	19.4	3131	6.8	226.4
手机旅行预订	7537	14.3	3493	7.5	115.8

1.1.3 Android 的特点

Android 有以下几个特点。

1. 开放性

首先是指 Android 从源码上开放,一个应用程序可以调用电话的任何核心功能;其次是指平台开放,不存在任何阻碍移动产业创新的专有权限制,任何联盟厂商都可自行定制基于 Android 操作系统的手机产品;第三是指运营上开放,手机使用什么方式接入什么网络用户可以自由选择,不再依赖运营商的控制。

2. 强大的应用开发平台

Android 提供的应用开发平台，使开发人员基于 Android 应用开发框架可以方便、灵活地开发各种移动应用。其核心应用和第三方应用完全平等。用户能完全根据自己喜好定制手机服务系统，并支持组件的重用和替换。应用程序可以轻松地嵌入网络功能支持，并可并行执行。

3. 支持丰富的硬件

由于 Android 的开放性，众多的厂商可推出各具功能特色的多种产品。

4. 巨大的市场前景

现在，使用 Android 手机操作系统的厂商有三星、摩托罗拉、索尼爱立信、HTC、LG 等，国内厂商有华为、联想、中兴等，厂商众多。

Android 市场占有率位居全球第一，机型数量庞大，简单易用，其开放性使厂商和客户可以定制桌面和主题风格，实现简单而华丽的界面。

5. 广泛的应用

Android 从手机操作系统开始，现已发展成为移动设备(如平板电脑、PDA)的操作系统，并进而成为物联网标准操作系统。

1.2　Android 操作系统的体系架构

Android 操作系统的体系架构包含 4 个层次，如图 1.1 所示。

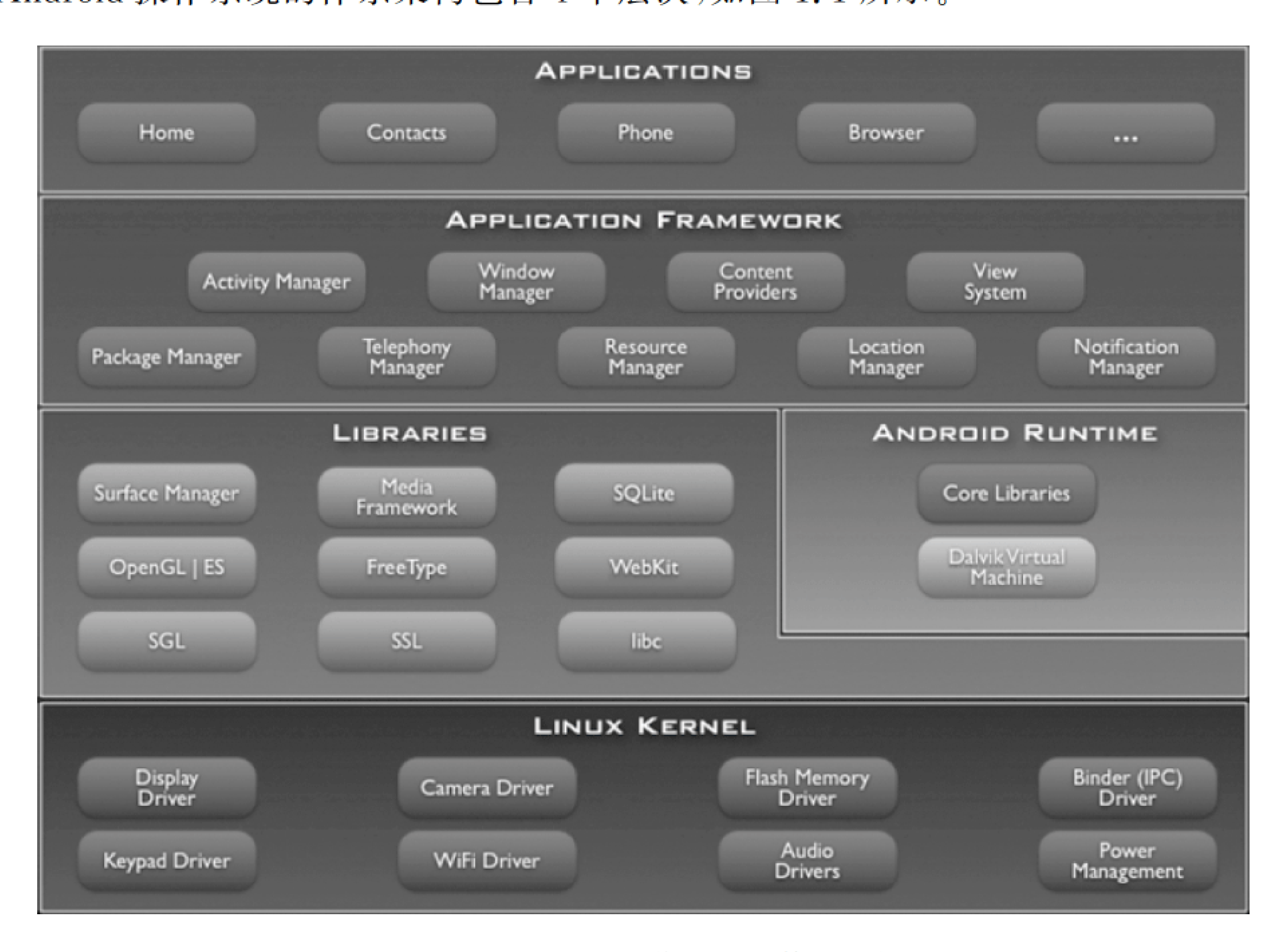


图 1.1　Android 操作系统的体系架构

由图 1.1 可以看出，Android 操作系统的体系架构由上到下依次为应用程序、应用程序框架、系统库和 Android 运行时、Linux 内核等 4 个层次，下面分别进行介绍。

1. 应用程序

在应用程序层，提供一系列的核心应用程序，例如电子邮件客户端、浏览器、通讯录、日历、地图、SMS 程序、联系人管理程序等，这些应用程序都是使用 Java 语言编写的。

用户可以用自己编写的应用程序来替换 Android 提供的应用程序，从而展示开发人员的才华并更具个性化。

2. 应用程序框架

应用程序框架层提供 Android 平台基本的管理功能和组件重用机制，它是从事 Android 开发的基础。该层提供了大量的 API 供开发人员使用，使开发人员易于开发功能强大的应用程序。

应用框架包含以下组件：

(1) Activity Manager(活动管理器)。管理应用程序的生命周期，起控制器的作用。

(2) Windows Manager(窗口管理器)。管理所有窗口程序。

(3) Content Provider(内容提供器)。提供一种服务，使应用程序之间可以实现数据共享，即使一个程序可以访问另一个程序的数据。

(4) View System(视图系统)。一组丰富并可扩展的视图组件，用于构建应用程序，例如文本框(TextView)、编辑框(EditText)、图片按钮(ImageButton)、复选按钮(Checkbox)等。

(5) Package Manager(包管理器)。管理安装在 Android 系统内的应用程序。

(6) Telephony Manager(电话管理器)。管理电话相关功能。

(7) Resource Manager(资源管理器)。对非编码资源进行统一管理。

(8) Location Manager(定位管理器)。管理地图相关服务功能。

(9) Notification Manager(通知管理器)。将应用消息显示在状态栏中，给用户以提示或通知。

3. 系统库和 Android 运行时

该层由系统库和 Android 运行时组成。

1) 系统库

系统库包含一套 C/C++库。系统库是应用程序框架的支撑，是连接应用程序框架层和 Linux 内核层的纽带，包含以下内容：

(1) Surface Manager。管理对显示子系统的访问，可对多个应用程序与 2D、3D 图形层提供无缝整合。

(2) Media Framework。Android 系统多媒体库，支持播放和录制许多流行的音频和视频格式。

(3) SQLite。所有应用程序都可使用的、功能强大的轻量级关系数据库引擎。

(4) OpenGL ES。3D 效果的支持。

(5) FreeType。位图和向量字体显示。

(6) WebKit。一个全新的 Web 浏览器引擎，支持 Android 浏览器和内嵌的 Web 视图。

(7) SGL。底层的 2D 图形引擎。

(8) SSL。为数据加密与安全传输提供支持。

(9) Libc。从 BSD 系统派生的标准 C 系统库,调整为适应嵌入式 Linux 设备。

2) Android 运行时

Android Runtime 为 Android 应用提供一个运行环境,包括核心库和 Dalvik 虚拟机。

(1) 核心库

核心库提供了 Java 编程语言核心类库具有的大部分功能。

(2) Dalvik 虚拟机

Dalvik 虚拟机是经过优化的多实例虚拟机,基于寄存器实现,采用专用的".dex"格式文件,该格式适合内存和处理器速度受限的系统。

每个 Android 应用程序都运行在单独的 Dalvik 虚拟机内,即每个 Android 应用程序都对应一个 Dalvik 进程。

Dalvik 虚拟机适合在移动终端上使用,相对于 PC 或服务器上的虚拟机来说,Dalvik 虚拟机能对内存高效使用,在低速 CPU 上表现出高性能,它有以下两个特点:

- 基于寄存器实现。Dalvik 虚拟机是基于寄存器的,而大多数虚拟机(例如 JVM)是基于栈的。
- 运行专有的.dex 文件。通过 DX 工具将应用程序所有.class 文件编译成.dex 文件,专有的.dex 文件减少了.class 文件的冗余信息,并用 DX 工具对.dex 文件进行优化,从而提高运行性能。

4. Linux 内核

Android 基于 Linux 2.6 内核,除了标准的 Linux 内核提供进程管理、内存管理、网络协议堆栈、驱动程序、安全机制等之外,Android 系统还增加了 Binder (IPC)驱动、Wi-Fi 驱动、蓝牙驱动等驱动程序,为系统运行提供了基础性支持。

Linux 内核部分是系统硬件和其他软件叠层之间的抽象层。

1.3 Eclipse 集成开发环境

开发 Android 应用程序,可在 Windows、Linux mac、Mac OS X 等平台上安装运行。本书使用 Windows 平台,搭建 Android 应用程序开发环境需要 JDK、Eclipse、ADT、SDK 等。

- JDK。Android 主要使用 Java 语言开发应用程序,需要使用 Java 软件开发包 JDK (Java 2 Software Development Kit)。
- Eclipse。Java 语言的集成开发环境。
- ADT。Android 使用 Eclipse 开发 Android 应用,并提供了专门的插件 ADT (Android Development Tools),使开发人员可以快速、方便地进行 Android 应用开发。
- SDK。SDK(Software Development Kit)是 Android 软件开发工具包。

1.3.1 JDK 下载和安装

在进行 Android 应用开发时,需要 Java SE 的支持,为方便软件开发的进行,需要安装

Java SE 开发环境 JDK(Java 2 Software Development Kit,Java 软件开发包)。

JDK 可以在 Oracle 公司的官方网站下载,网址如下: http://www.oracle.com/technetwork/java/index.html。

在浏览器地址栏中输入该地址后,可以看到 Java SE SDK 的下载版本,本书以当前流行版本 Java SE7。

本书在 Windows 平台下进行开发,必须下载 Windows 版本,下载之后得到的可执行文件为 jdk-7u45-windows-i586。

双击下载后的安装文件 jdk-7u45-windows-i586,出现"许可证"窗口后,单击"接受"按钮。在"自定义安装"窗口中,使用默认选项,单击"下一步"按钮,即可进行安装。安装目录是 C:\Program Files (x86)\Java\jdk1.7.0_45。

通过设置系统环境变量,告诉 Windows 操作系统 JDK 的安装位置,环境变量设置方法如下。

(1) 设置系统变量 Path。在"开始"菜单中,选择"控制面板"→"系统"→"高级系统设置"→"环境变量",出现如图 1.2 所示的"环境变量"对话框。在"系统变量"中找到变量名为 Path 的变量,单击"编辑"按钮,弹出"编辑系统变量"对话框,在"变量值"文本框中输入 JDK 的安装路径: C:\Program Files (x86)\Java\jdk1.7.0_45\bin,如图 1.3 所示,单击"确定"按钮完成配置。



图 1.2 "环境变量"对话框

(2) 设置系统变量 JAVA_HOME。在"系统变量"中单击"新建"按钮,弹出"编辑用户变量"对话框,在"变量名"文本框中输入 JAVA_HOME,在"变量值"文本框中输入 JDK 的安装路径: C:\Program Files (x86)\Java\jdk1.7.0_45,如图 1.4 所示,单击"确定"按钮完成配置。

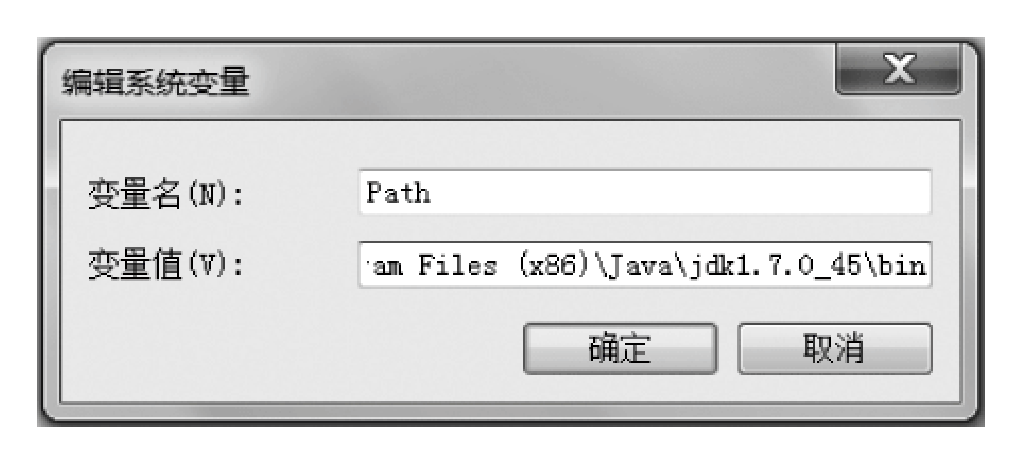



图 1.3　编辑变量 Path

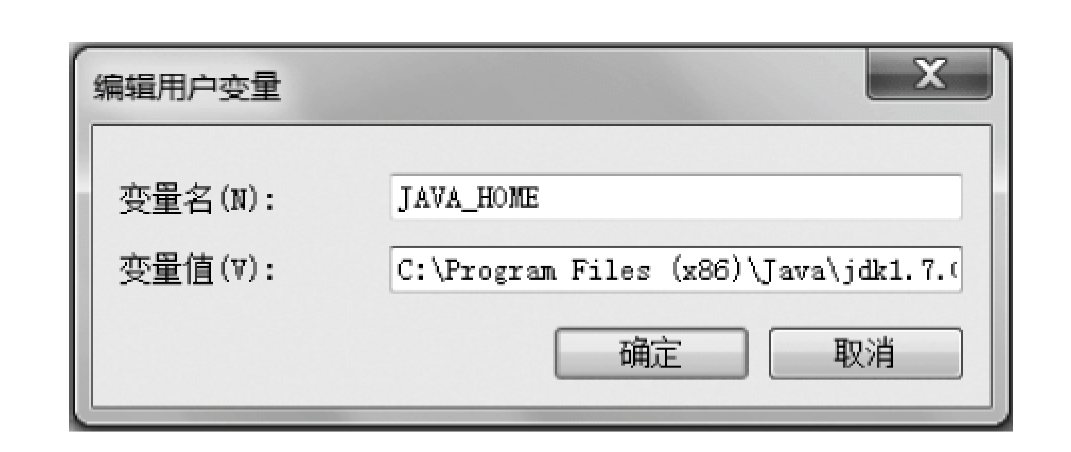


图 1.4　新建变量 JAVA_HOME

1.3.2　Eclipse 集成开发环境的下载与安装

Android 的 SDK，可在网址 http://developer.android.com/sdlk/index.html 下载，选择合适的 Android SDK 版本。

为了快速搭建 Android 集成开发环境，使用 Android 提供的一个集成 Eclipse、ADT 和 SDK 的 Android SDK 版本，文件名为 adt-bundle-windows-x86-20140321.zip；下载完毕后，将该文件解压到指定安装位置即可，本书的安装位置为 E:\adt-bundle-windows-x86-20140321。解压后的文件夹中包括文件夹 eclipse、sdk 和应用程序 SDK Manager.exe，如图 1.5 所示。

本书以此版本为例介绍 Android 集成开发环境的搭建和应用程序的开发。

在图 1.5 所示的 eclipse 文件夹中，双击文件 eclipse.exe，Eclipse 集成开发环境启动画面如图 1.6 所示。



图 1.5　adt-bundle-windows-86-20140321 解压后的文件夹



图 1.6　Eclipse 启动画面

1.3.3　Eclipse 集成开发环境的界面

Eclipse 集成开发环境启动完成后的界面如图 1.7 所示。

在图 1.7 所示的界面中，从上到下包含菜单栏、工具栏、各种面板和状态栏。左侧的 Package Explorer 面板和中部的代码编辑器面板是开发程序常用的面板。在 Package Explorer 面板中显示所有 Android 项目的目录树，在目录树中双击某个文件，即可在代码编辑器中显示文件内容，并可对其内容进行编辑修改。

1. 菜单栏

在 Android 开发环境界面顶部第 2 行是菜单栏，包含主菜单（例如 File，Edit，…，

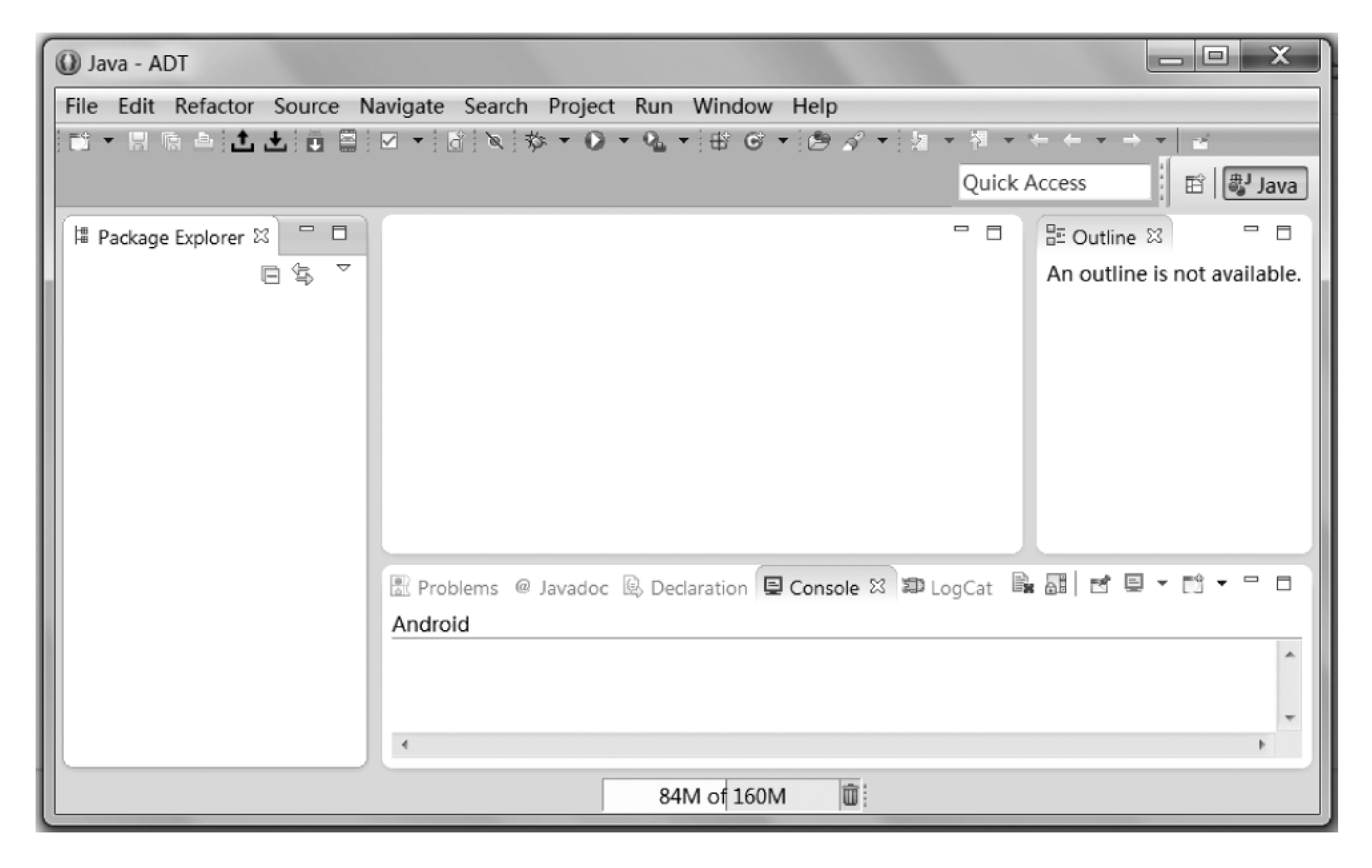


图 1.7　Eclipse 开发环境界面

Window，Help)和其所属的菜单项，菜单项下面还可以有子菜单，例如选择 File→New，出现的菜单如图 1.8 所示。

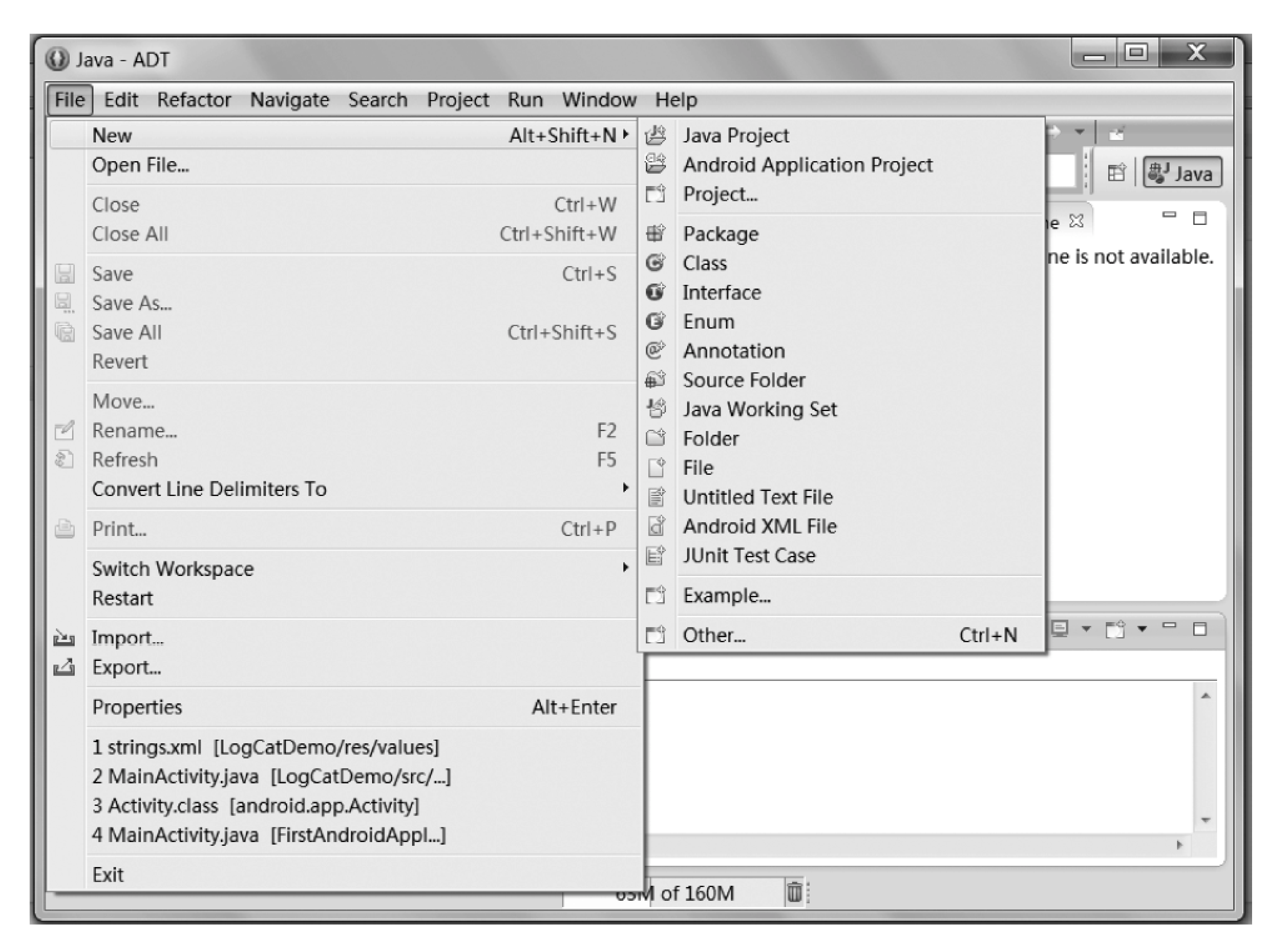


图 1.8　Eclipse 开发环境界面的菜单栏

2. 工具栏

菜单栏下面两行是工具栏,如图 1.9 所示。

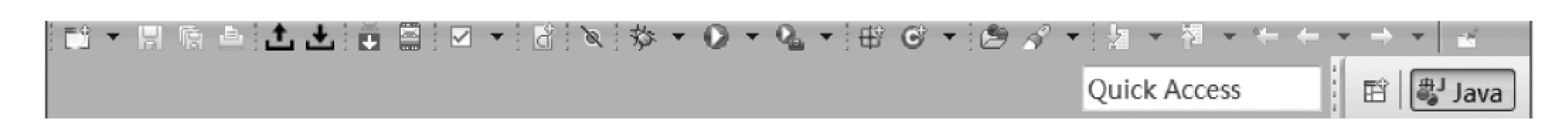


图 1.9 Eclipse 开发环境界面的工具栏

3. 透视图

位于工具栏第 2 行右侧的第 2 个图标是"打开透视图",它可以显示多个透视图以供切换,单击"打开透视图"图标,出现"打开透视图"窗口,如图 1.10 所示。

一个透视图相当于一个自定义的界面,它保存了当前的菜单栏、工具栏以及透视的大小、位置等,可在下次切换时恢复原来的布局。

如果需要恢复到默认的 Android 开发环境的界面,可选择其中的 Java (default)选项。

4. 视图

视图是主界面中的一个小窗口,可以调整显示大小、位置或关闭,也可以最大化、最小化。Android 开发环境的界面是由工具栏、菜单栏、状态栏和许多视图构成的,选择 Window→Show,出现显示视图菜单,如图 1.11 所示。

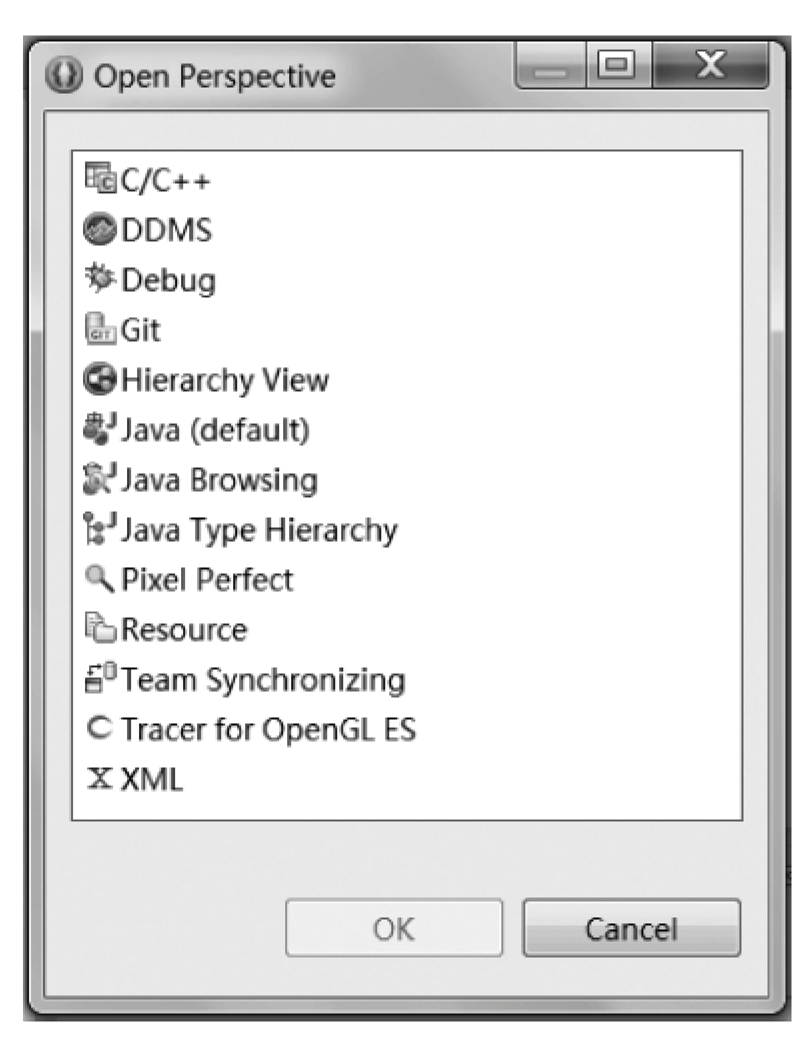


图 1.10 "打开透视图"窗口

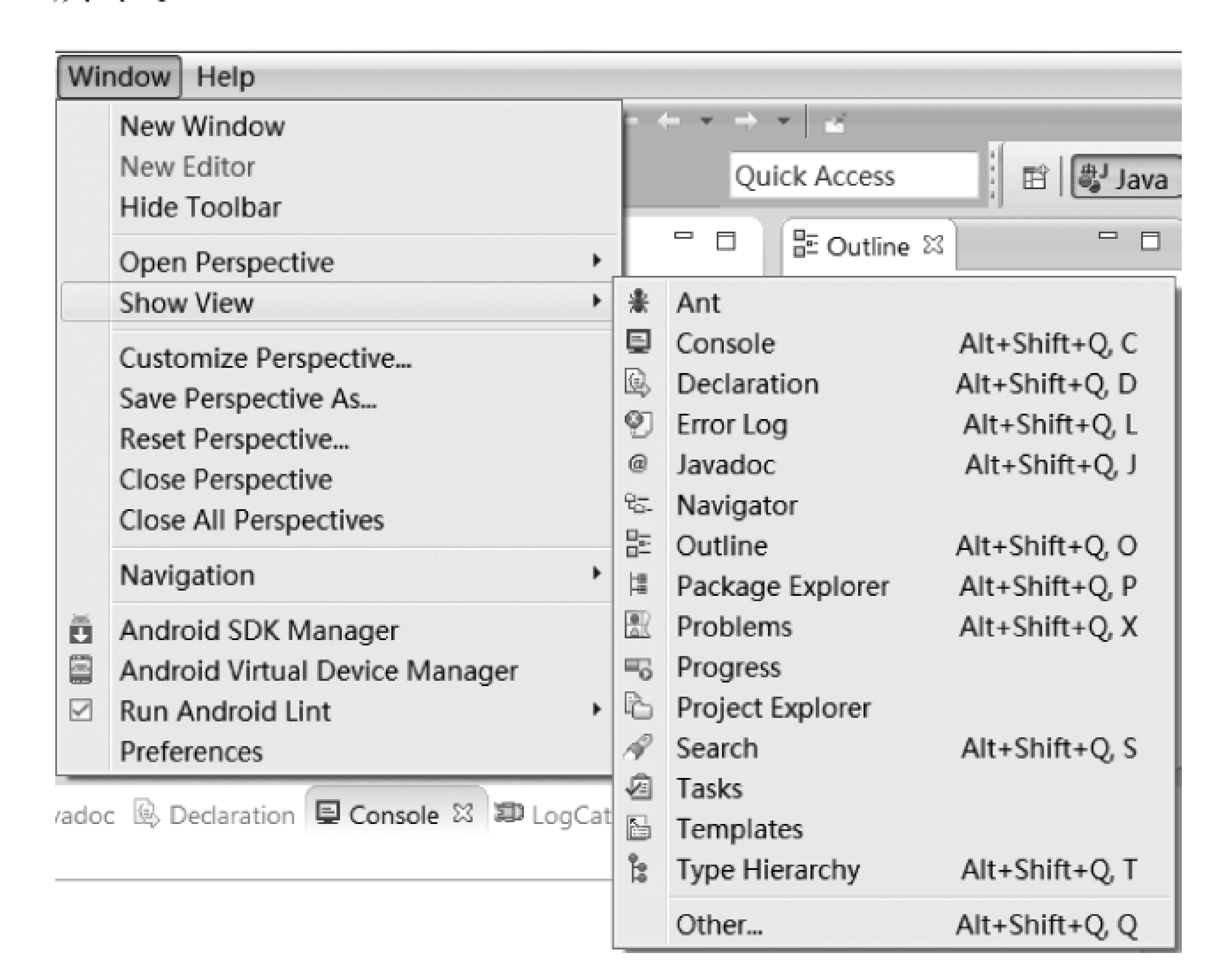


图 1.11 显示视图菜单

5. 代码编辑器

代码编辑器在界面的中间,用于编辑程序代码,并具有自动调试和排错功能。该编辑器与视图相似,也能最大化和最小化,如图 1.12 所示。

6. 设置 SDK 和 JDK 的路径

在 Android 集成开发环境安装完成后,系统会自动完成设置 SDK 和 JDK 的路径。如果没有自动完成上述工作,或需要进行版本升级,可以手动设置。

```
MainActivity.java

package com.application.firstandroidapplication;

import android.app.Activity;
import android.app.ActionBar;
import android.app.Fragment;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.view.ViewGroup;
import android.os.Build;

public class MainActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        if (savedInstanceState == null) {
            getFragmentManager().beginTransaction()
                    .add(R.id.container, new PlaceholderFragment()).commit();
        }
    }

    @Override
```

图 1.12　代码编辑器

1）设置 SDK 的路径

在图 1.11 中，选择菜单 Window→Preference，出现 Preference 对话框，选择左边目录树中的 Android，此时右边窗口显示 Android Preference 的内容，如图 1.13 所示。

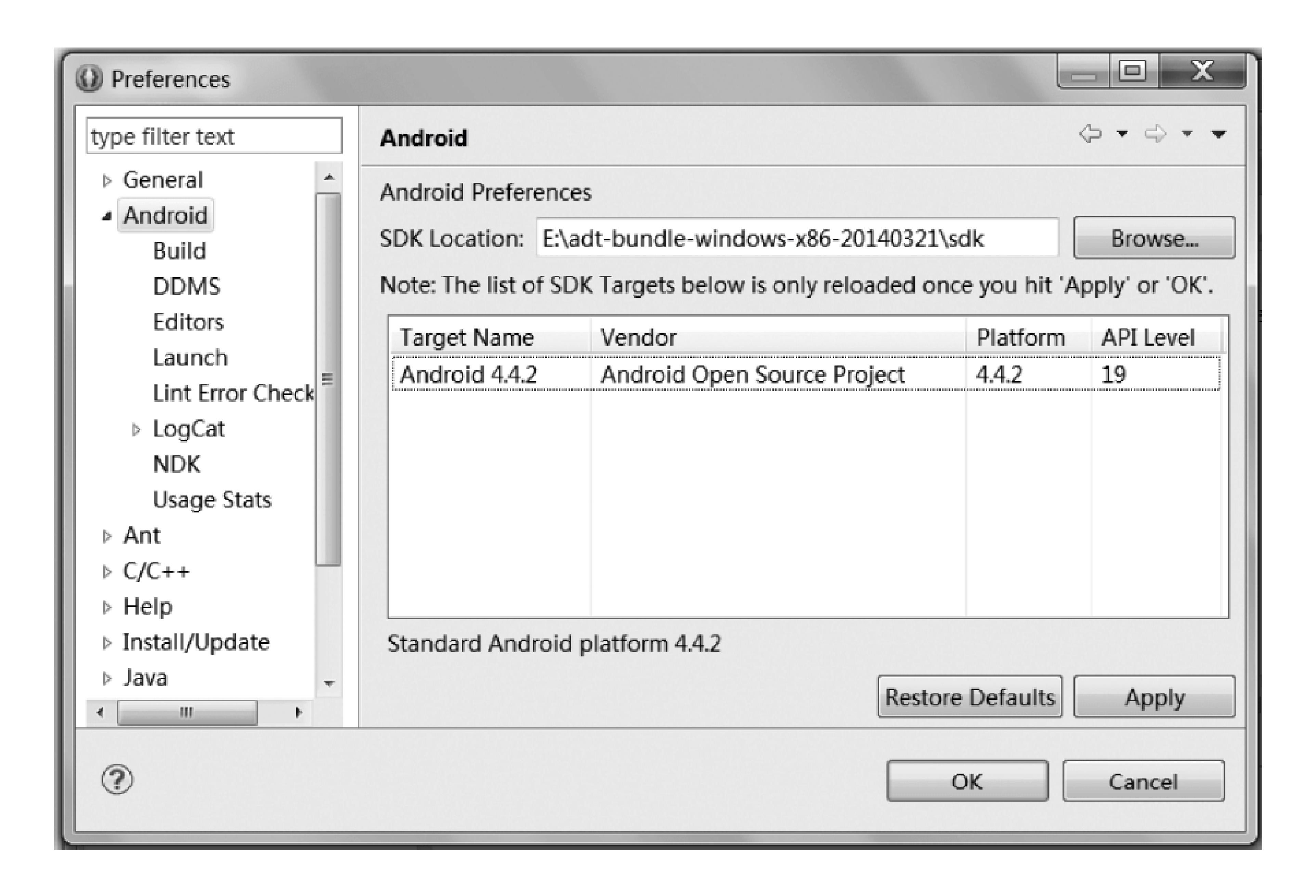


图 1.13　设置 SDK

可以单击 Browse…按钮选择 SDK 所在文件夹的路径，也可在 SDK Location 文本框中输入 SDK 所在文件夹的路径。

2）设置 JDK 的路径

在图 1.11 中，选择菜单 Window→Preference，出现 Preference 对话框，选择左边目录树中的 Java→Install JREs，此时右边窗口显示 Installed JREs 的内容，如图 1.14 所示。

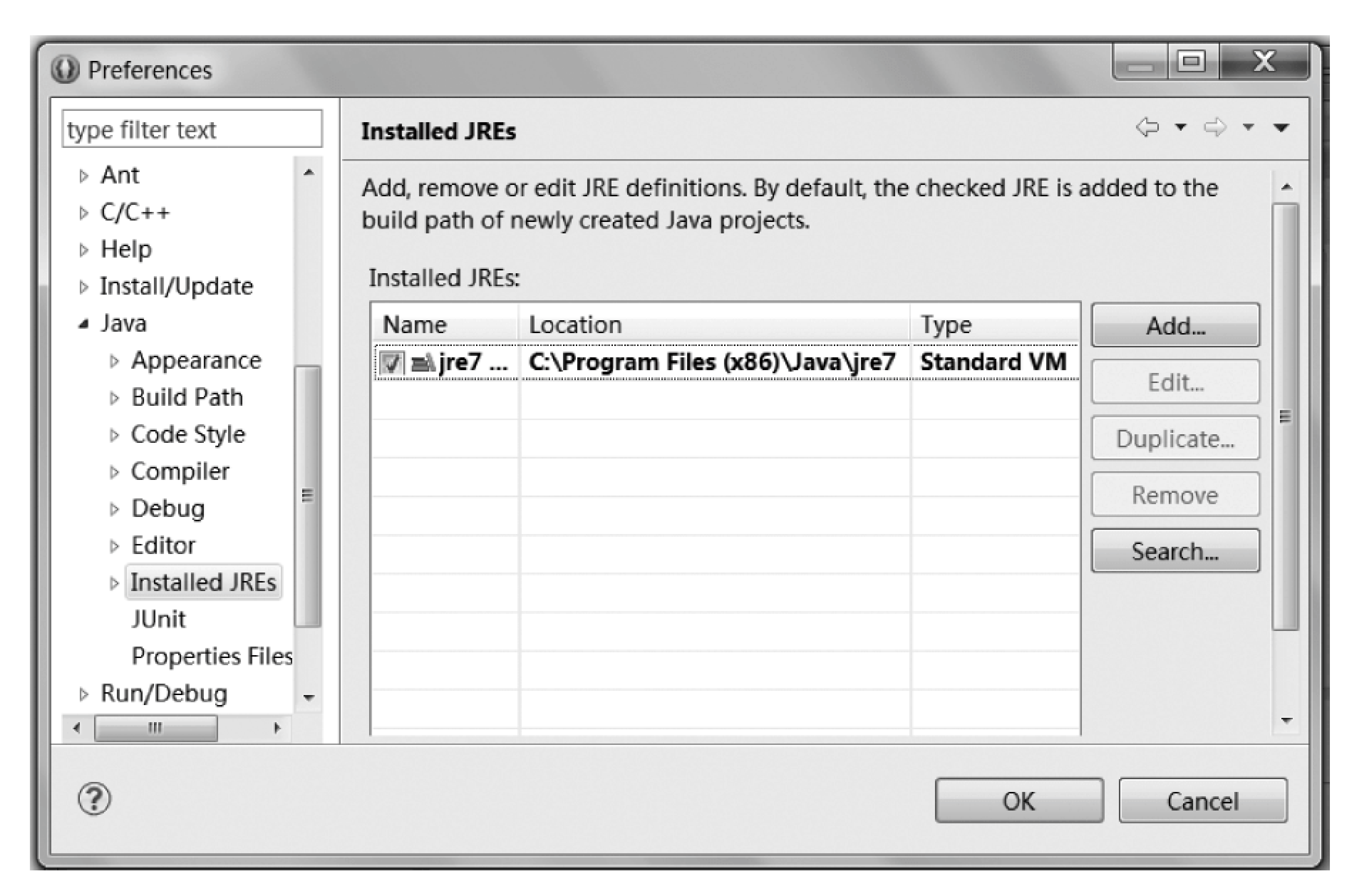


图 1.14　设置 JDK

可以单击 Add…按钮指定 JRE 所在文件夹的路径。

1.3.4　创建和启动虚拟设备 AVD

在 Eclipse 集成开发环境中，创建和启动 Android 虚拟设备（模拟器）Android Virtual Device(AVD)的步骤如下。

（1）在 Eclipse 中，选择菜单 Window→Android Virtual Device Manager，出现 Android Virtual Device Manager 对话框，此时还没有任何模拟器在对话框的列表中。

（2）单击 New…按钮，进入 Create new Android Virtual Device（AVD）对话框，在 AVD Name 框中创建一个新的模拟器，设置名称为 AVDTest 。在 Device 下拉列表框中，选择 HVGA。在 Target 下拉列表框中，选择 Android4.4.2-API Level 19，如图 1.15 所示。

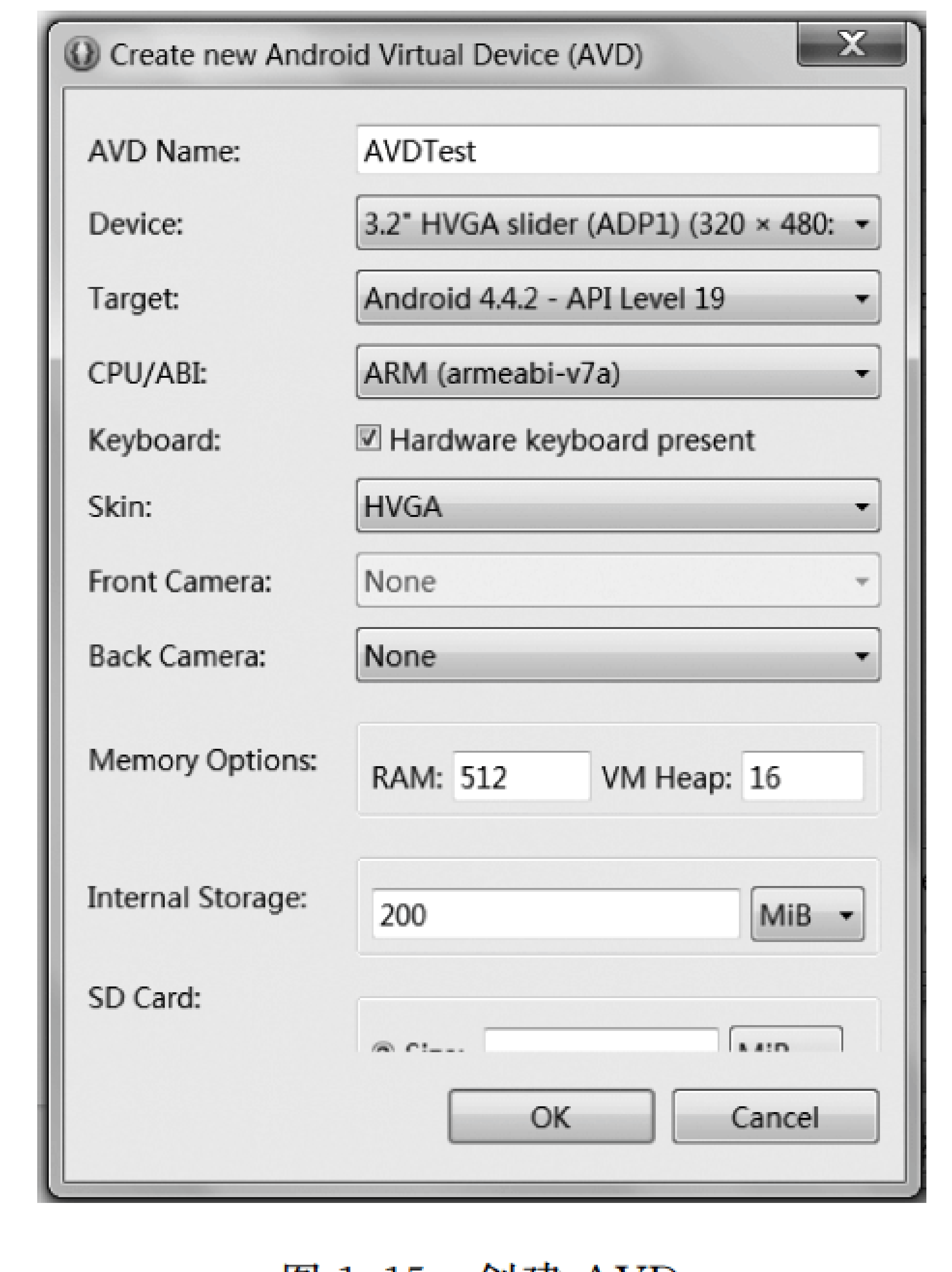


图 1.15　创建 AVD

单击 OK 按钮,完成新的模拟器的创建。

(3) 在如图 1.16 所示的 Android Virtual Device Manager 对话框中,选择已创建的模拟器 AVDTest,单击 Start 按钮。

出现 Launch Options 对话框,如图 1.17 所示,单击 Launch 按钮。

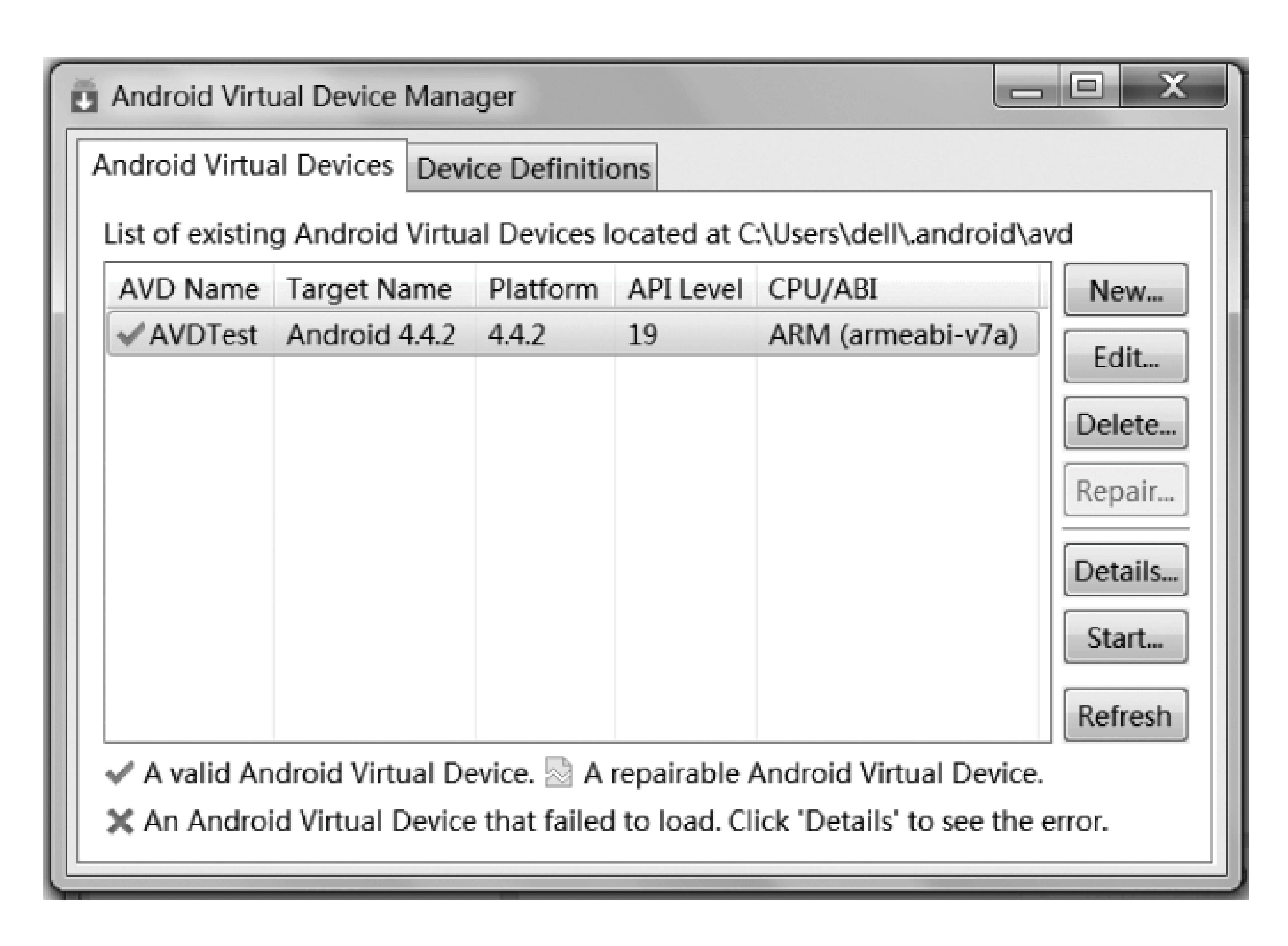


图 1.16　Android Virtual Device Manager 对话框

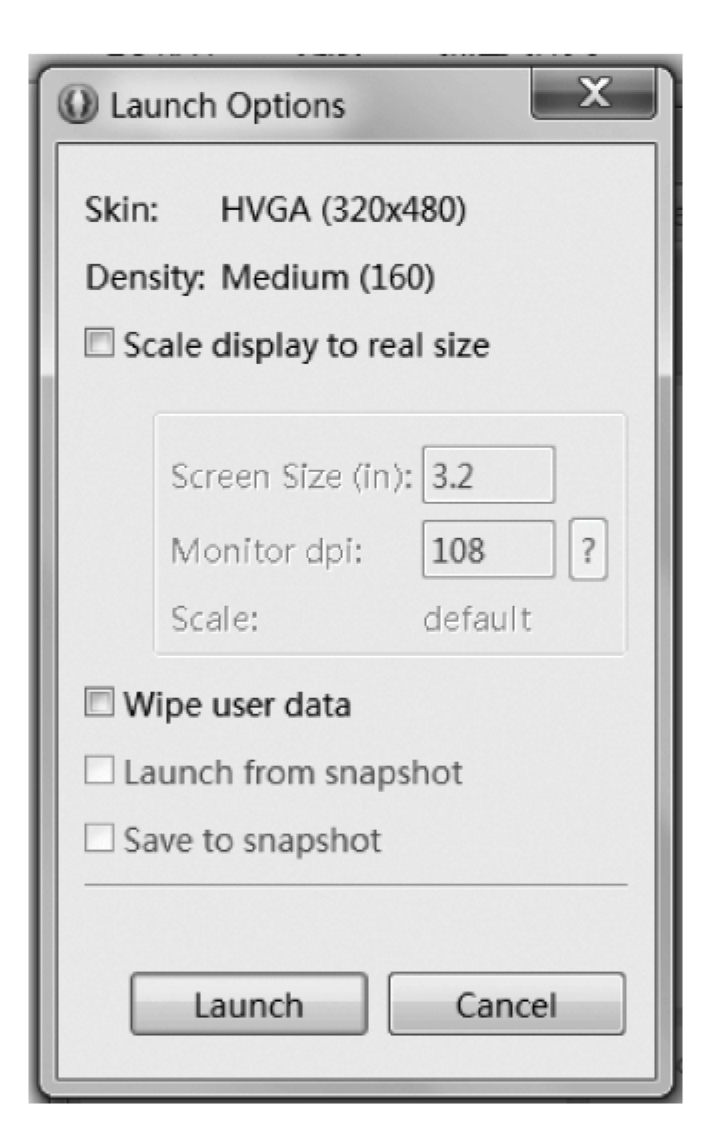


图 1.17　Launch Options 对话框

(4) 启动运行 Android 模拟器,如图 1.18 所示。

图 1.18　Android 模拟器

启动后的 Android 模拟器与真手机很相似。它由手机屏幕和键盘按钮面板两部分构成，手机屏幕在模拟器的左部，键盘按钮面板在模拟器的右部。

1. 手机屏幕

图 1.18 左部显示手机桌面，可以通过鼠标在上面单击模拟手点击手机屏幕，底部的 5 个按钮功能如下：

(Calling 键)：呼叫电话。

(Contacts 键)：单击显示联系人。

(APPS 键)：显示应用程序。

(Messaging 键)：收发短信。

(Browser 键)：浏览网页。

2. 键盘按钮面板

键盘按钮面板主要键的功能如下：

：Camera 键。

：减小音量键。

：增加音量键。

：关闭电源键。

：Call 键。

：Hang Up 键。

：Home 键。

：Menu 键。

：Back 键。

：Search 键。

1.4 Android Studio 集成开发环境

Android Studio 是 Google 为开发设计人员提供的最新集成开发环境，基于优秀的 IntelliJ IDEA 工具，除 IntelliJ 工具外，Android Studio 还提供基于 Gradle 的构建支持。Google 建议开发设计人员尽快从 Eclipse 集成开发环境转为使用 Android Studio 集成开发环境。

安装 Android Studio 需要 JDK 7 或以上的版本。

下载 Android Studio 的网址为 http://developer.android.com/sdk/index.html，也可以从网站 http://www.androiddevtools.cn/下载。

为了搭建 Android Studio 集成开发环境，下载压缩包为 android-studio-ide-141.2456560-windows，下载完毕后，将该文件解压到指定安装位置即可，本书的安装位置为 D:\Studio\android-studio-ide-141.2456560-windows。

单击解压路径下的 bin 目录，studio64.exe 文件，直接运行安装。

安装完成后，启动 Android Studio，出现 Android Studio 启动界面，如图 1.19 所示。出现 Android Studio 的开始对话框，如图 1.20 所示。



图 1.19　Android Studio 启动界面

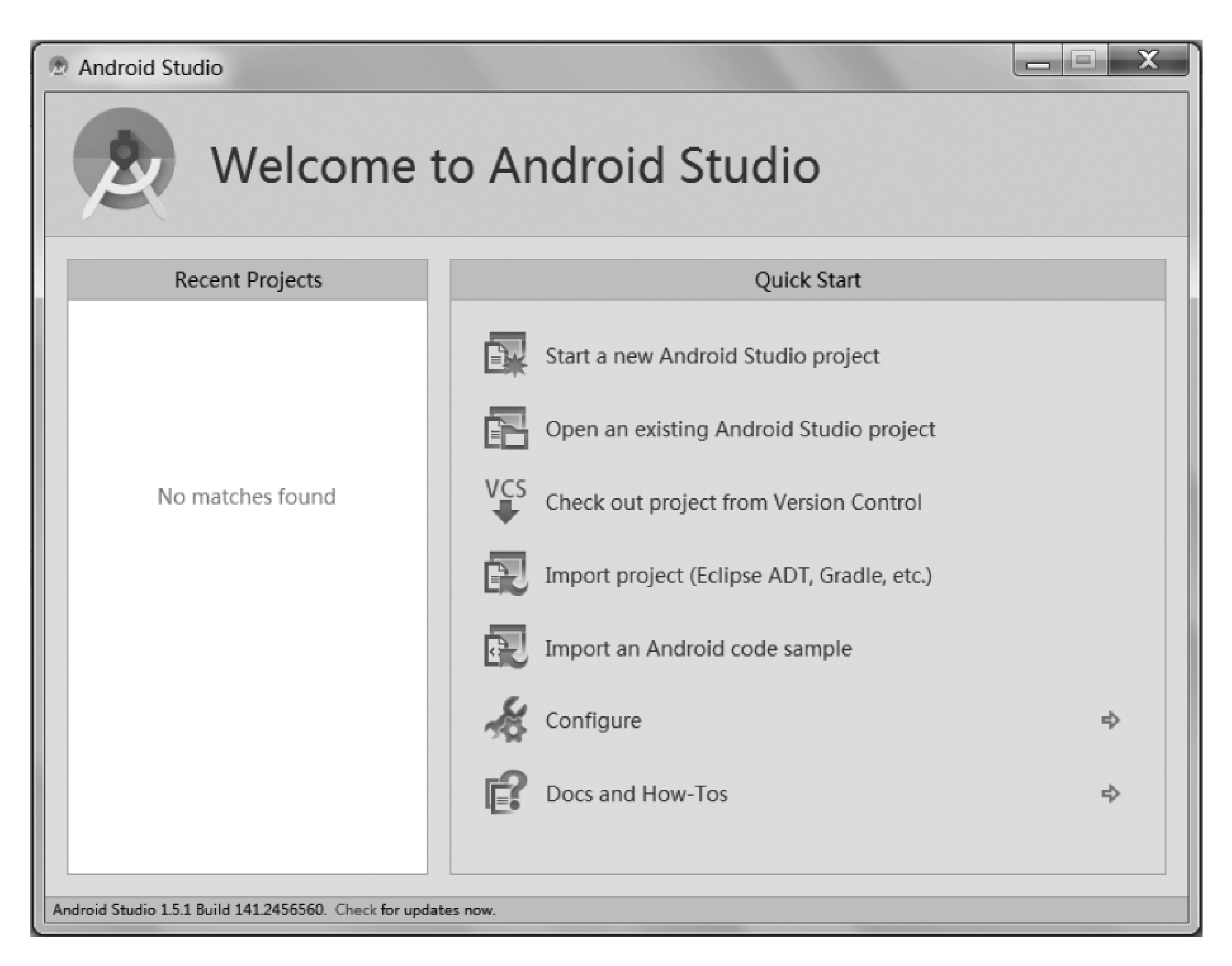


图 1.20　Android Studio 的开始对话框

选择图 1.20 中的 Start a new Android Studio project，出现新建项目对话框，如图 1.21 所示。

在图 1.21 中，在 Application name 框中输入“FirstApplication”，在 Company Domain 框中输入“application.com”，在 Package name 框中输入“com.application.firstapplication”，在 Project Location 框中输入“C:\Users\dell\Desktop\FirstApplication”，单击 Next 按钮，出现选择 Android SDK 版本，如图 1.22 所示。

在图 1.22 中，选择 SDK 最低版本，这里选择 API 19 Android 4.4，单击 Next 按钮，出现添加 Activity 对话框，如图 1.23 所示。

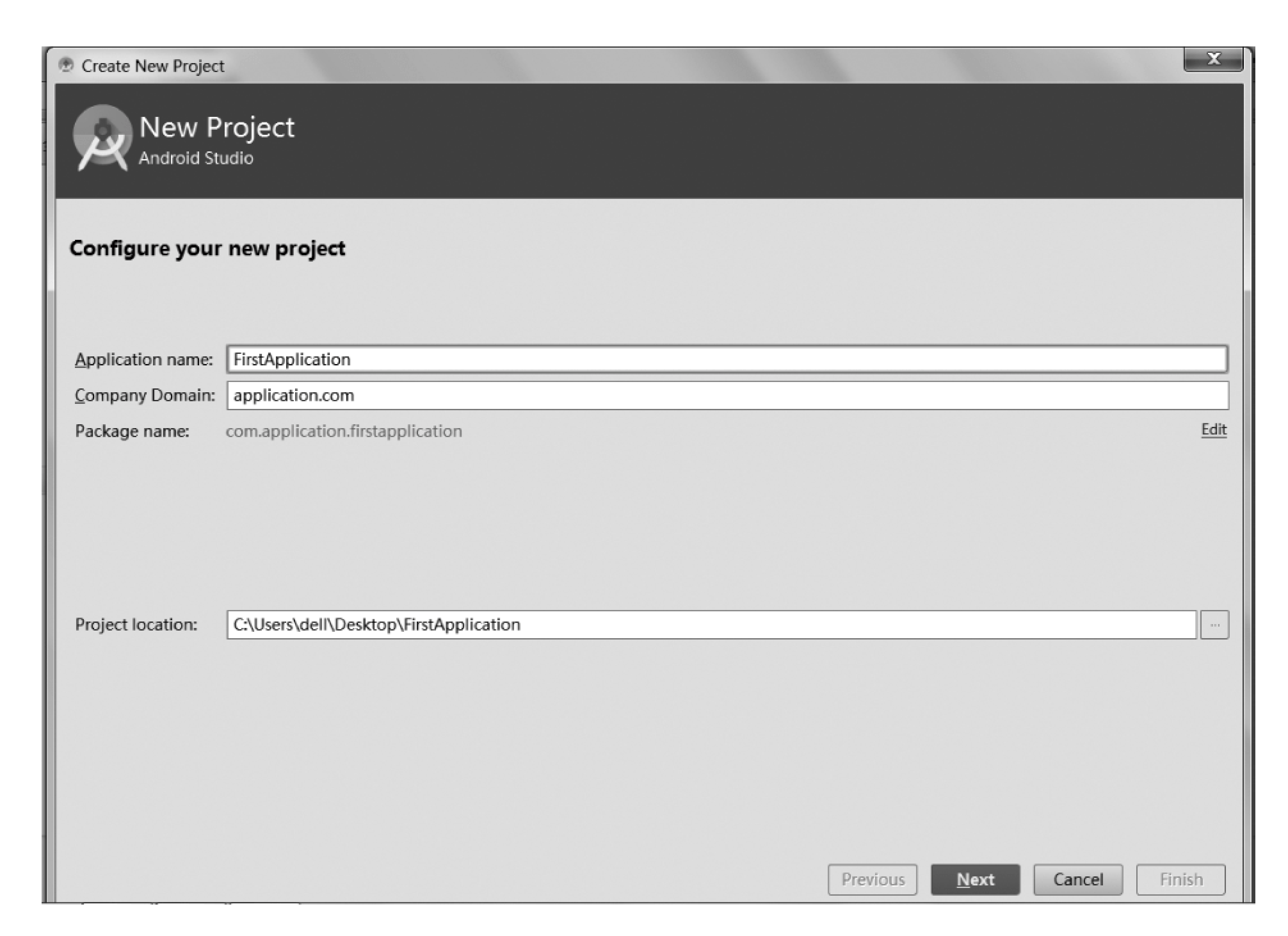


图 1.21　新建项目对话框

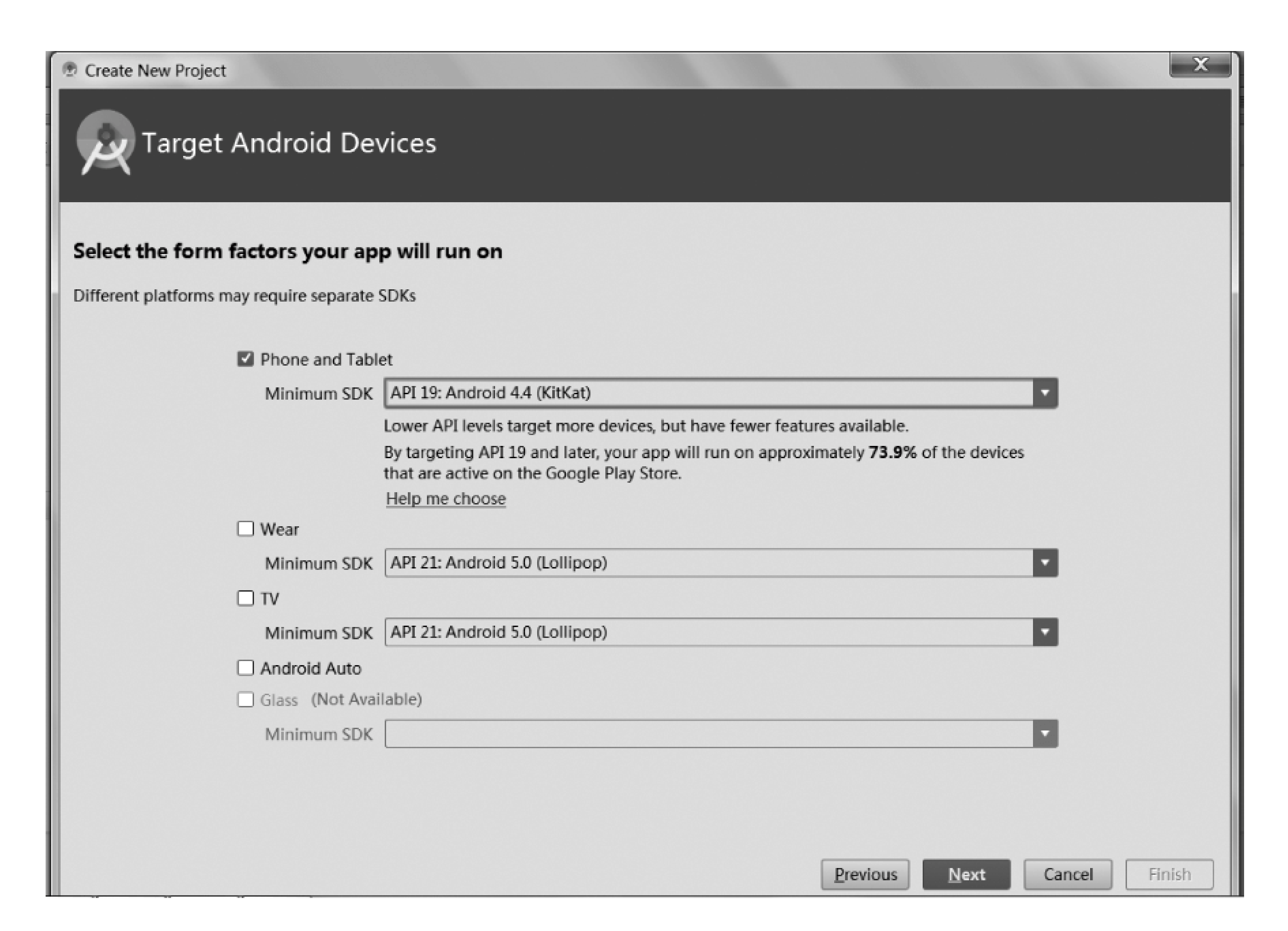


图 1.22　选择 Android SDK 版本

在图 1.23 中，选择 Blank Activity，单击 Next 按钮，出现设置 Activity 名称，如图 1.24 所示。

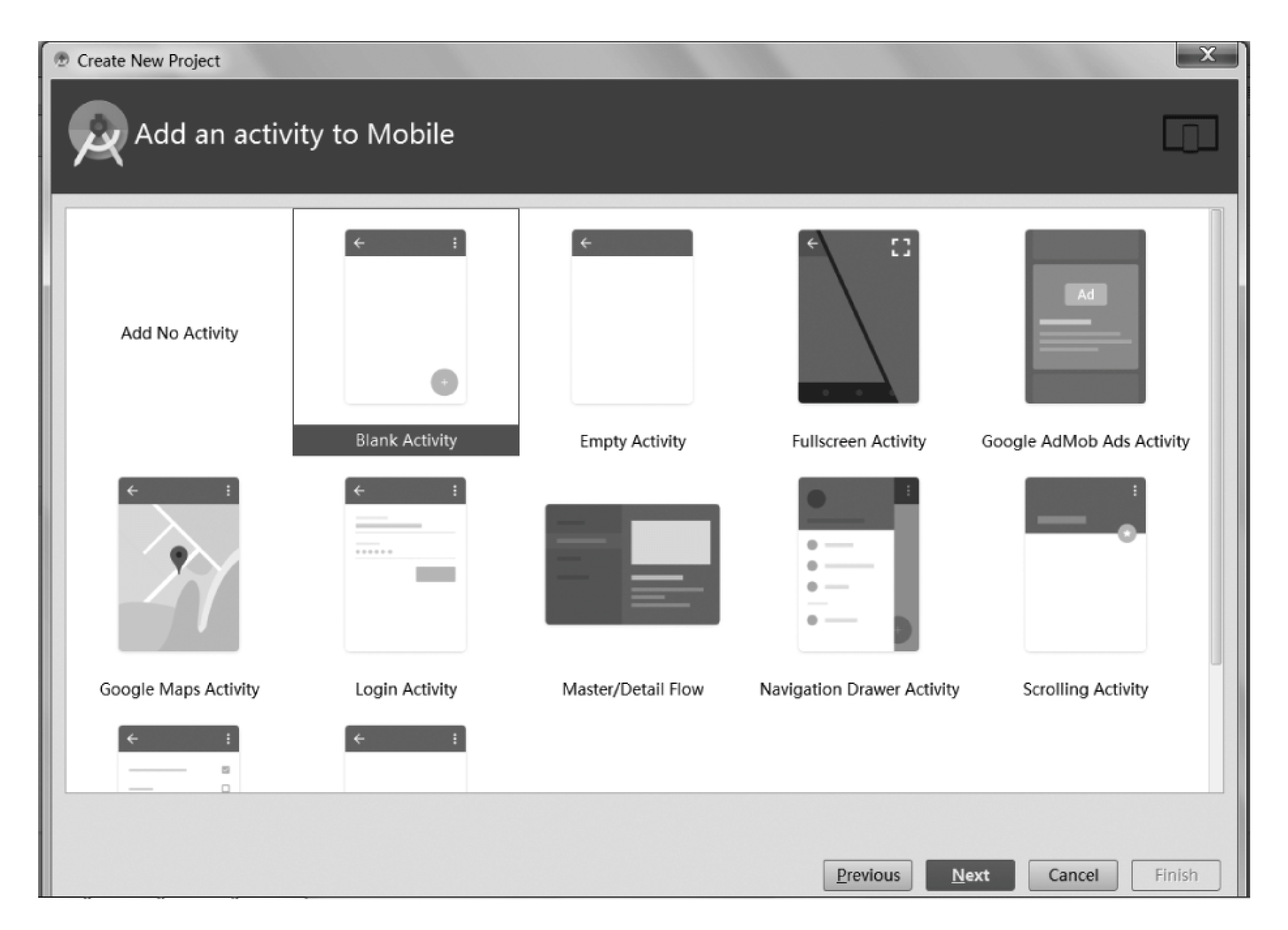


图 1.23　添加 Activity 对话框

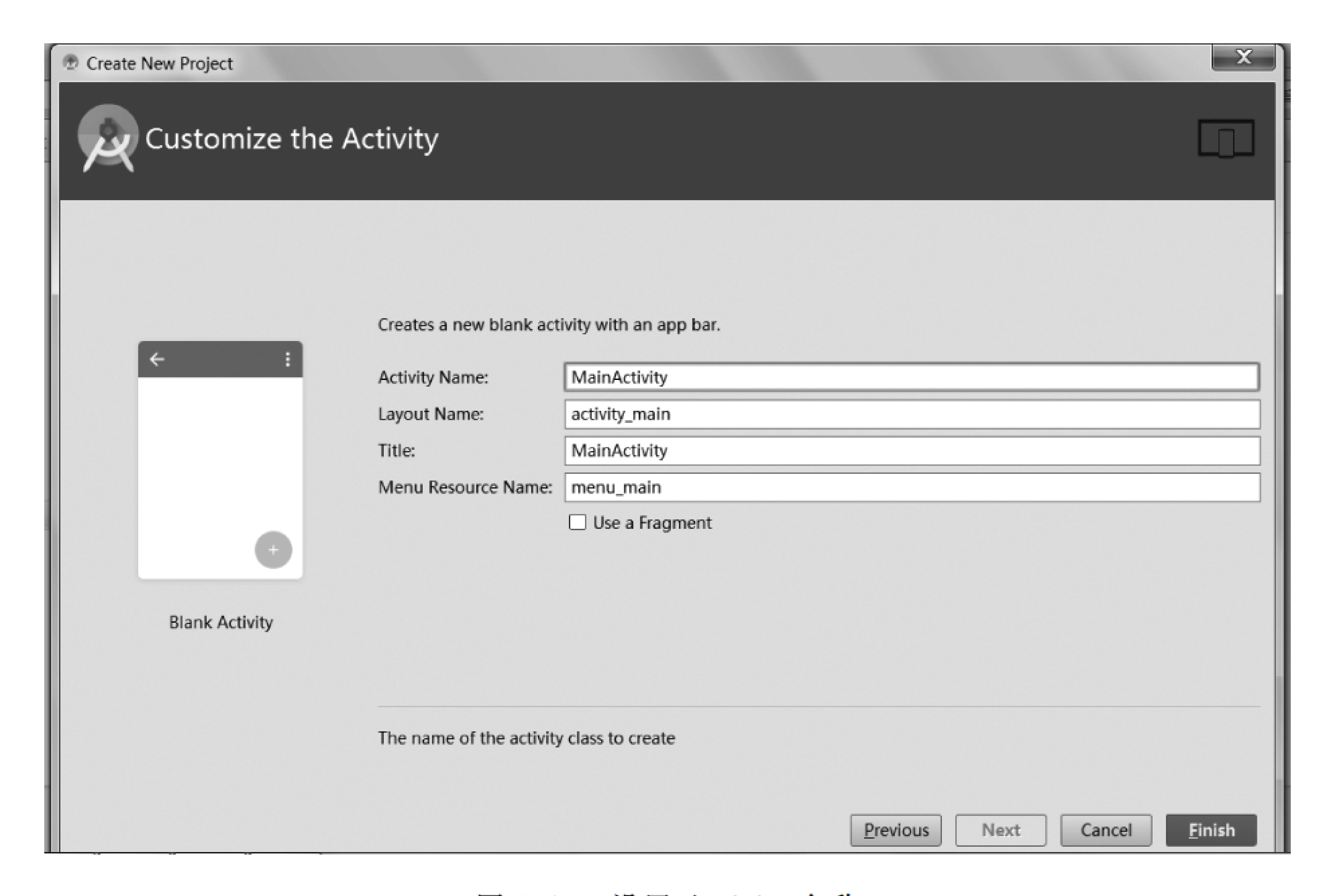


图 1.24　设置 Activity 名称

在图 1.24 中，在 Activity Name 框中输入“MainActivity”，单击 Finish 按钮，出现 Android Studio 集成开发环境的窗口，如图 1.25 所示。

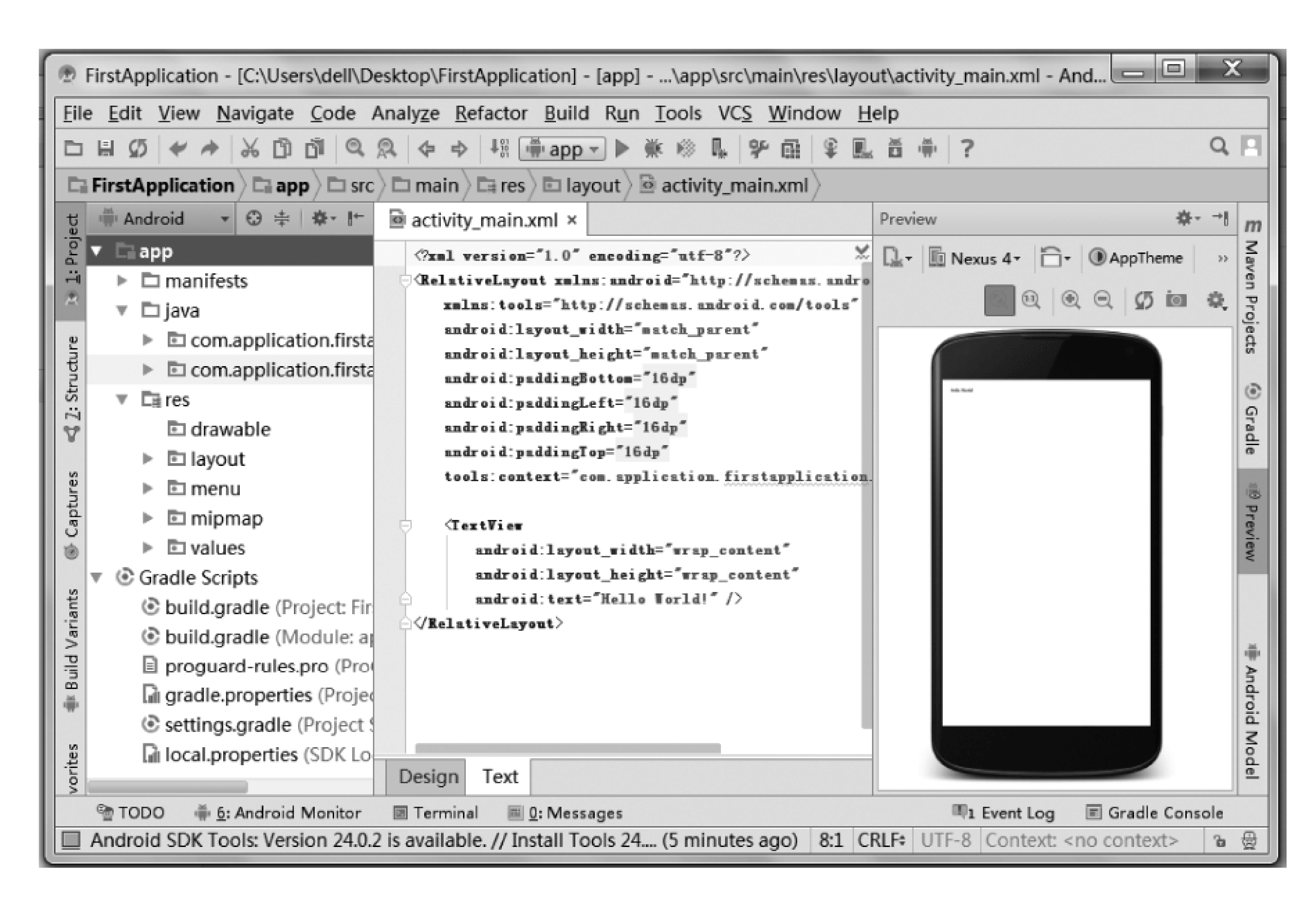


图 1.25　Android Studio 集成开发环境的窗口

单击工具栏 AVD Manager 按钮，在出现的对话框中单击 Create Virtual Device…按钮，即可创建新的 Android 模拟器，与 Eclipse 创建的 Android 模拟器界面相同。

单击工具栏 SDK Manager 按钮，出现 Android SDK 设置对话框，如图 1.26 所示，可选择 Android SDK 新版本，例如 Android 6.0 或 Android 7.0，进行版本升级。

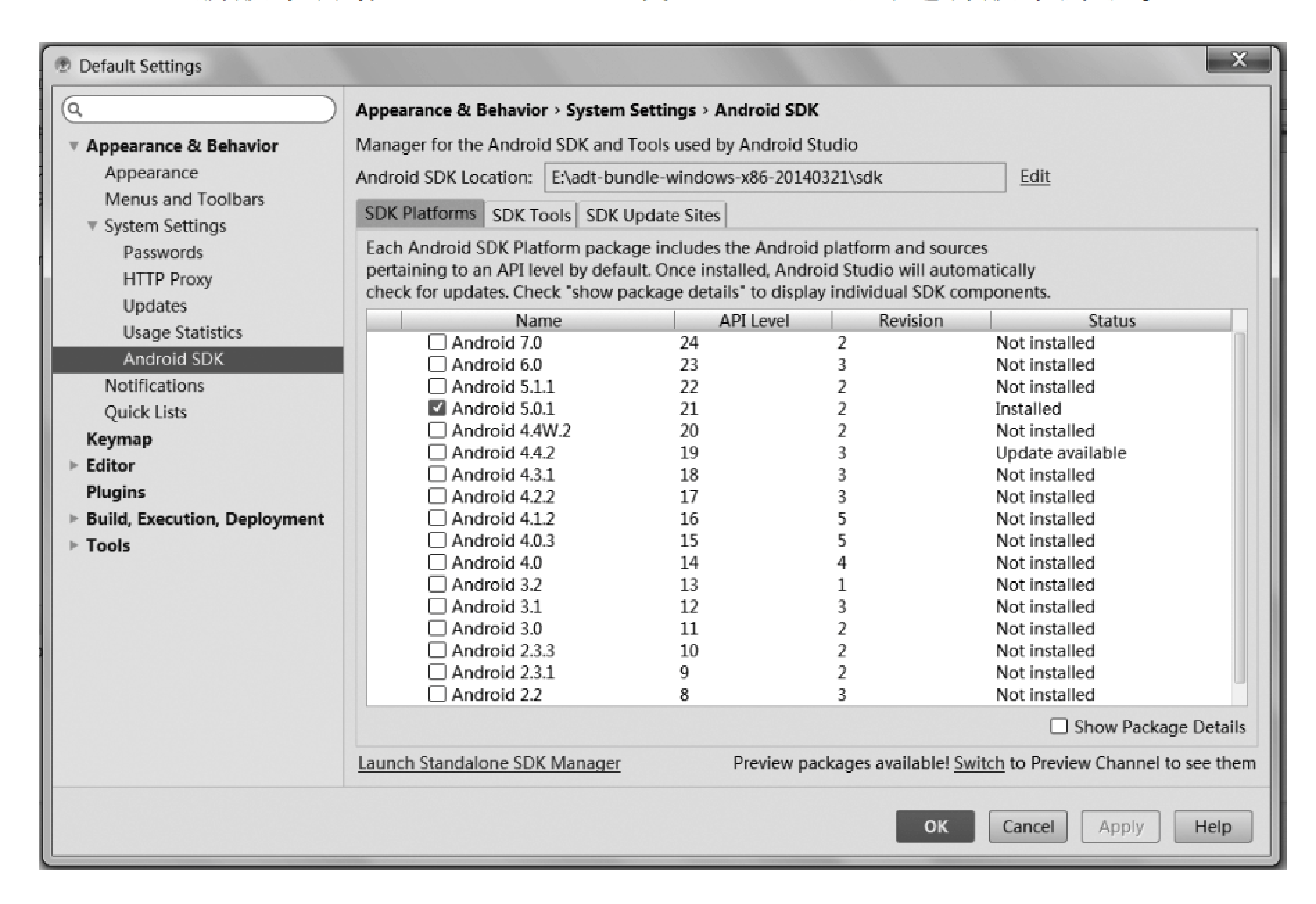


图 1.26　Android SDK 设置对话框

1.5 小　　结

本章主要介绍了以下内容：

(1) Android 具有开放性、强大的应用开发平台、支持丰富的硬件、巨大的市场前景、广泛的应用等特点。

(2) Android 系统平台可以开发出通信、搜索、新闻、娱乐、商务、家电控制、传感器、可穿戴设备等移动服务应用，以及进行 Android 与物联网结合的应用开发。中国手机网民各类手机网络应用情况表明：移动应用对人们的信息、社交、娱乐和购物等各方面产生重要影响，其中，电子商务类应用和娱乐类应用尤为突出，并通过手机打车、手机地图和手机支付等应用加大对社会生活服务的渗透。

(3) Android 操作系统的体系架构包括应用程序、应用程序框架、系统库和 Android 运行时、Linux 内核等 4 个层次。

- 应用程序层。提供一系列的核心应用程序，例如电子邮件客户端、浏览器、通讯录、日历、地图、SMS 程序、联系人管理程序等，这些应用程序都是使用 Java 语言编写的。
- 应用程序框架层。提供 Android 平台基本的管理功能和组件重用机制，它是从事 Android 开发的基础，应用框架包含以下组件：Activity Manager(活动管理器)、Windows Manager(窗口管理器)、Content Provider(内容提供器)、View System(视图系统)、Package Manager(包管理器)、Telephony Manager(电话管理器)、Resource Manager(资源管理器)、Location Manager(定位管理器)、Notification Manager(通知管理器)等。
- 系统库和 Android 运行时层。系统库包含一套 C/C++库，它是应用程序框架的支撑，是连接应用程序框架层和 Linux 内核层的纽带。Android 运行时为 Android 应用提供一个运行环境，包括核心库和 Dalvik 虚拟机。
- Linux 内核层。Android 基于 Linux 2.6 内核，除了标准的 Linux 内核提供进程管理、内存管理、网络协议堆栈、驱动程序、安全机制等之外，Android 系统还增加了 Binder(IPC)驱动、Wi-Fi 驱动、蓝牙驱动等驱动程序，为系统运行提供了基础性支持。

(4) Eclipse 集成开发环境。

- JDK 下载和安装。在进行 Android 应用开发时，需要安装 Java SE 开发环境 JDK，本书的安装目录是 C:\Program Files\Java\jdk1.7.0_67。
- Android 集成开发环境的下载与安装。为了快速搭建 Android 集成开发环境，使用 Android 提供的一个集成 Eclipse、ADT 和 SDK 的 Android SDK 版本，文件名为 adt-bundle-windows-x86-20140321.zip，本书的安装位置为 E:\adt-bundle-windows-x86-20140321。
- Android 集成开发环境的界面与项目的创建、导入、导出、移除和运行。
- 创建和启动 Android 虚拟设备 AVD。

(5) Android Studio 集成开发环境。Android Studio 是 Google 为开发设计人员提供的最新集成开发环境，基于优秀的 IntelliJ IDEA 工具。除 IntelliJ 工具外，Android Studio 还提供基于 Gradle 的构建支持。Google 建议开发设计人员尽快从 Eclipse 集成开发环境转为使用 Android Studio 集成开发环境。

为了搭建 Android Studio 集成开发环境，下载压缩包为 android-studio-ide-141.2456560-windows.zip；下载完毕后，将该文件解压到指定安装位置即可，本书的安装位置为 D:\Studio\android-studio-ide-141.2456560-windows。

习 题 1

一、选择题

1.1 应用程序框架层不包含的组件是________。

A. Activity Manager　　B. Content Provider

C. View System　　D. Web Services

1.2 系统库未包含的库是________。

A. SQLite　　B. Math　　C. WebKit　　D. FreeType

二、填空题

1.3 Android 具有开放性、________、支持丰富的硬件、巨大的市场前景、广泛的应用等特点。

1.4 Android 系统平台可以开发出通信、搜索、新闻、娱乐、________、家电控制、传感器、可穿戴设备等移动服务应用。

1.5 Android 系统还增加了 Binder (IPC)驱动、________、蓝牙驱动等驱动程序。

1.6 搭建 Eclipse 应用程序开发环境需要 JDK、Eclipse、ADT、________等。

1.7 为了快速搭建 Eclipsed 集成开发环境，使用 Android 提供的一个集成 Eclipse、SDK 和________的 Android SDK 版本，文件名为 adt-bundle-windows-x86-20140321.zip。

1.8 Android Studio 集成开发环境基于优秀的________工具。

三、问答题

1.9 简述 Android 系统的特点。

1.10 简述 Android 系统的应用。

1.11 Android 操作系统的体系架构包括哪几个层次？各层有何特点？

四、应用题

1.12 分别下载、安装和配置 JDK 1.7、adt-bundle-windows-x86-20140321.zip，搭建 Eclipse 集成开发环境。

1.13 分别下载、安装和配置 JDK 1.7、android-studio-ide-141.2456560-windows.zip，搭建 Android Studio 集成开发环境。

1.14 进行 Android 项目的导入、导出和移除等上机实验。

1.15 创建和启动 Android 虚拟设备 AVD。

第 2 章 Android 应用的创建、调试和发布

本章要点

- 创建第一个 Android 应用项目。
- 在模拟器或移动设备上运行 Android 应用程序。
- 应用项目的导入、导出和移除。
- Android 应用的目录结构包括 src 目录、gen 目录、bin 目录、res 目录和 AndroidManifest.xml 文件等。
- 对应用项目的源代码文件、资源文件、资源索引文件和项目配置文件的代码进行分析。
- Android 应用调试工具有 Java 调试器 Debug、图形化调试工具 DDMS 和获取日志信息调试工具 LogCat。
- 运行 Android 应用程序的过程有编译、打包、安装和运行。

在搭建 Android 集成开发环境后，就可以进入 Android 应用项目的开发，本章介绍 Android 应用项目的创建和运行、目录结构、程序分析、调试和发布等内容。

2.1 Android 项目的创建和运行

应用 Android 开发平台的结构框架，创建一个新的 Android 应用项目是十分简捷、方便的，本节介绍 Android 应用项目的创建、运行、导入、导出和移除。

2.1.1 创建第一个 Android 应用项目

下面举例说明 Android 应用项目的创建。

【例 2.1】 创建第一个 Android 应用项目 FirstAndroidApplication。

创建应用项目步骤如下：

(1) 启动 Eclipse，在 Android 开发环境界面中，选择菜单 File→New→Android Application Project，或在工具栏中执行 New→Android Application Project 命令，显示创建 Android 应用项目对话框，如图 2.1 所示。

在该对话框中填写以下内容：

- Application Name：应用名，这里填入“FirstAndroidApplication”。应用名应遵守文件夹命名规则，不能使用中文名。
- Project Name：项目名与应用名是一致的，这里也是“FirstAndroidApplication”，在填写 Application Name 后会自动填入。

- Package Name：应用程序的包名，这里填入“com. application. firstandroidapplication”。

提示：Android 应用程序的包名十分重要，必须是唯一的，包名可作为 Android 应用的唯一标识。

- Minimum Required SDK：运行应用程序最低的 SDK 版本号，这里是 API14，Android SDK 4.0。
- Target SDK：运行应用程序的目标 SDK 版本号，这里是 API19，Android SDK 4.4。
- Compile With：编译应用程序的 SDK 版本号，这里是 API19，Android SDK 4.4。

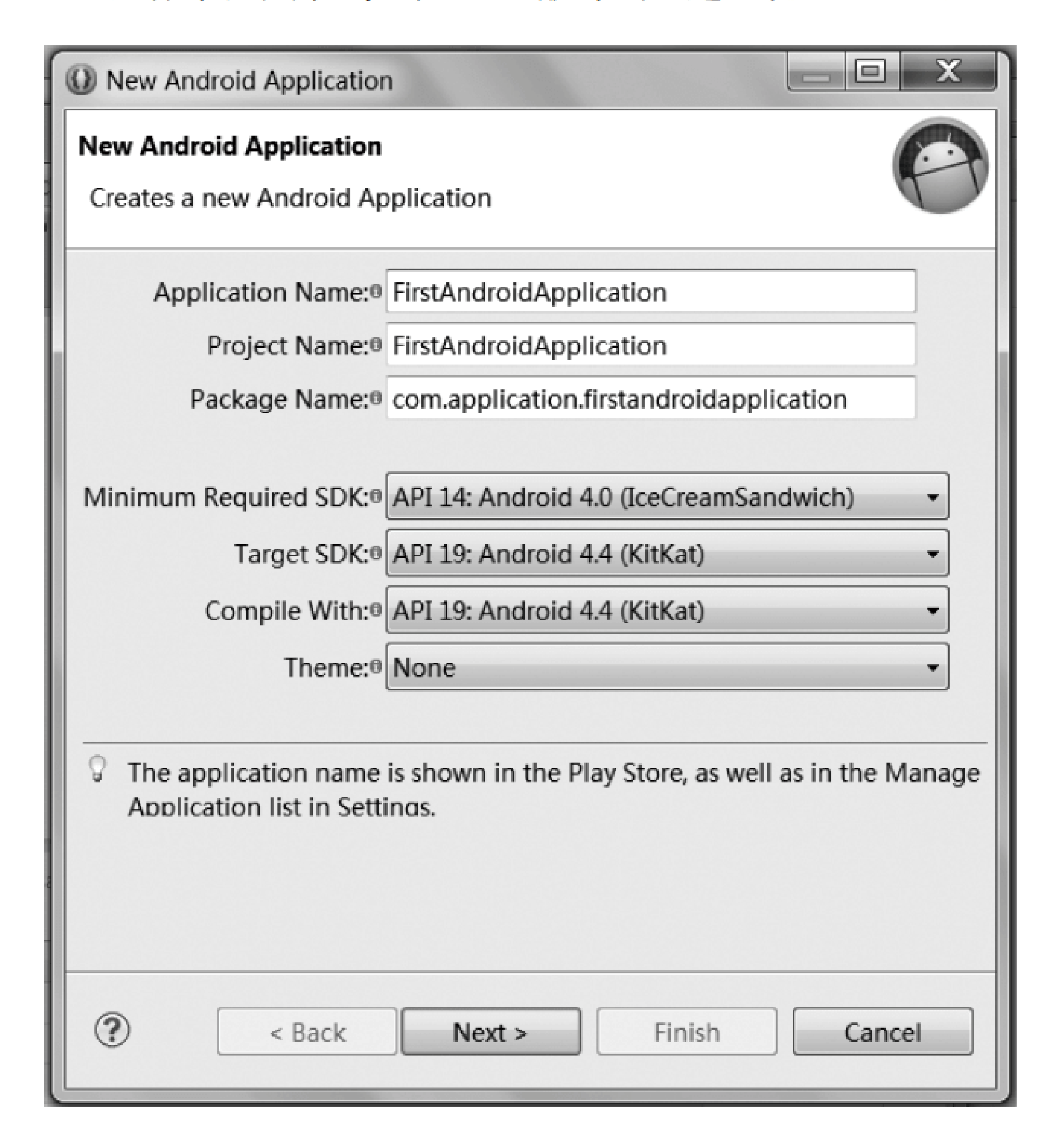


图 2.1　创建 Android 应用项目对话框

(2) 单击 Next 按钮，在出现的对话框中设置应用程序的启动图标和 Activity，如图 2.2 所示。

在图 2.2 中：

- Create activity 选项：用于创建 Activity，本例选择了该选项。
- Create custom launcher icon 选项：如果选择了该选项，可在接下来的对话框中设置应用程序的图标，否则会跳过这个对话框，采用默认的图标。本例选择了该选项。

(3) 单击 Next 按钮，弹出如图 2.3 所示的对话框，可设置应用程序的图标，这里采用默认设置。

(4) 单击 Next 按钮，弹出如图 2.4 所示的对话框，选择一个 Activity 的模板，本例选择 Blank Activity。

(5) 单击 Next 按钮，弹出如图 2.5 所示的对话框，用于设置 Activity 的名称，本例采用默认设置。单击 Finish 按钮，完成应用项目的创建。

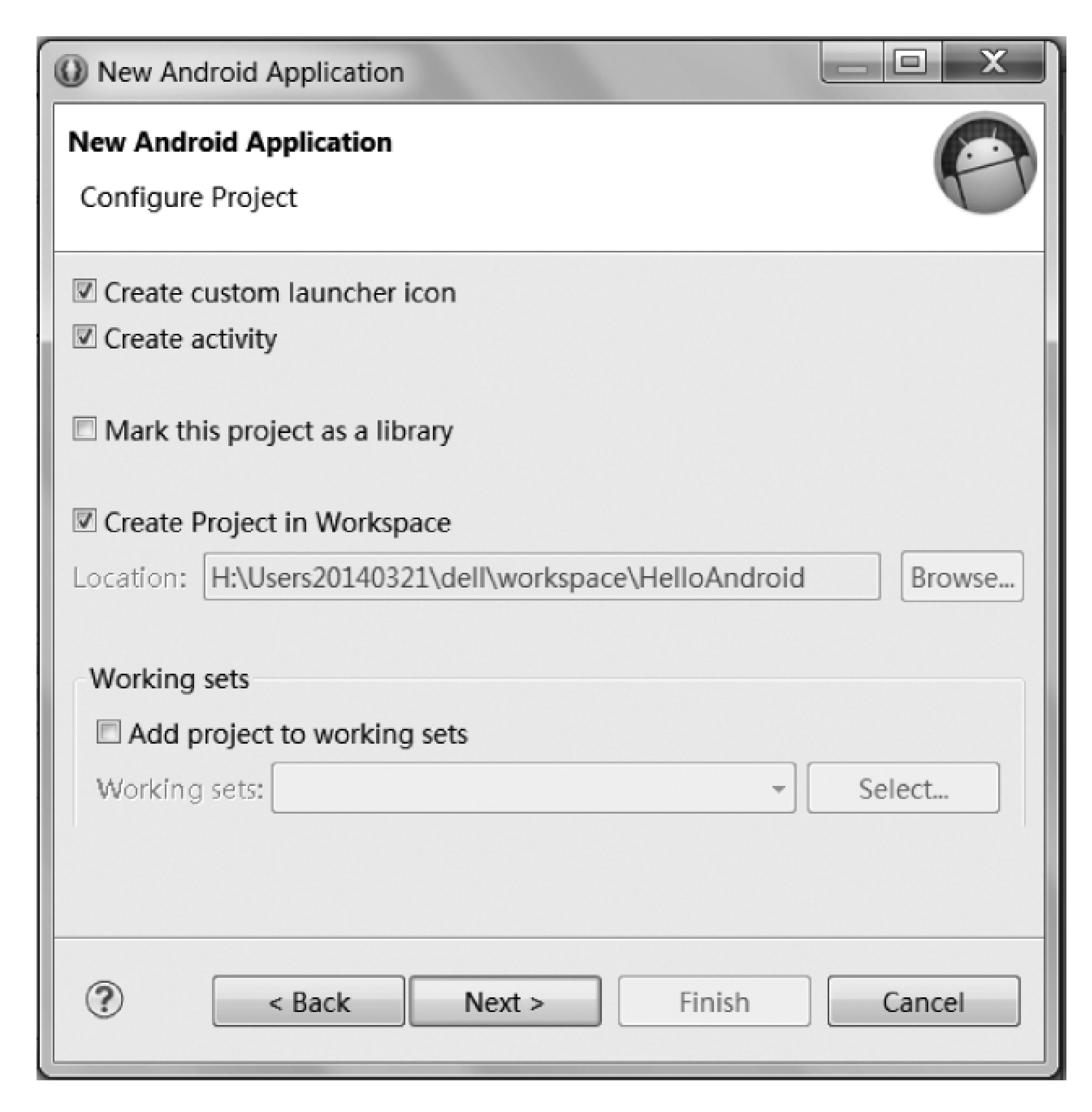


图 2.2　设置应用程序的启动图标和 Activity

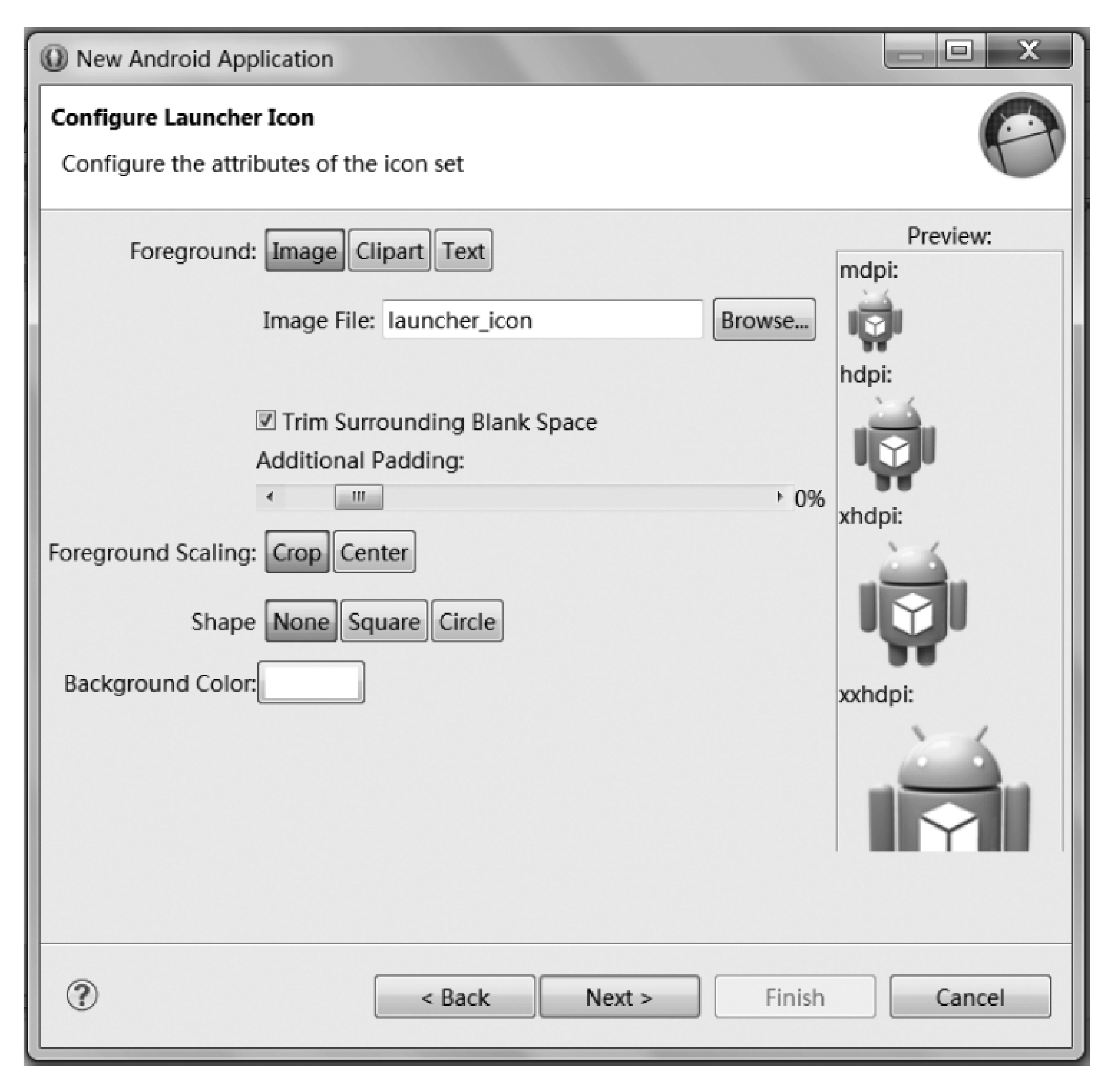


图 2.3　设置应用程序的图标

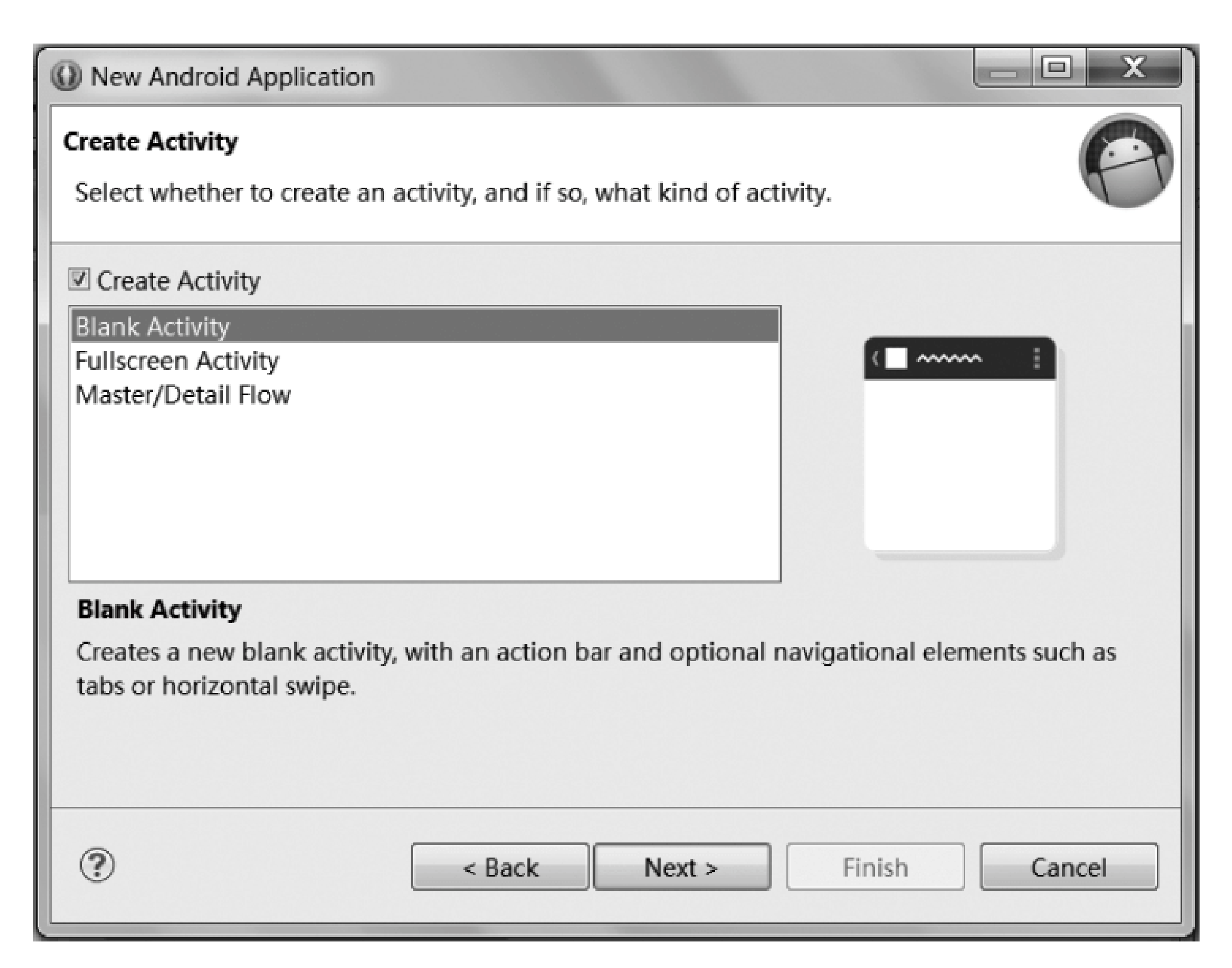


图 2.4　选择 Activity 的模板

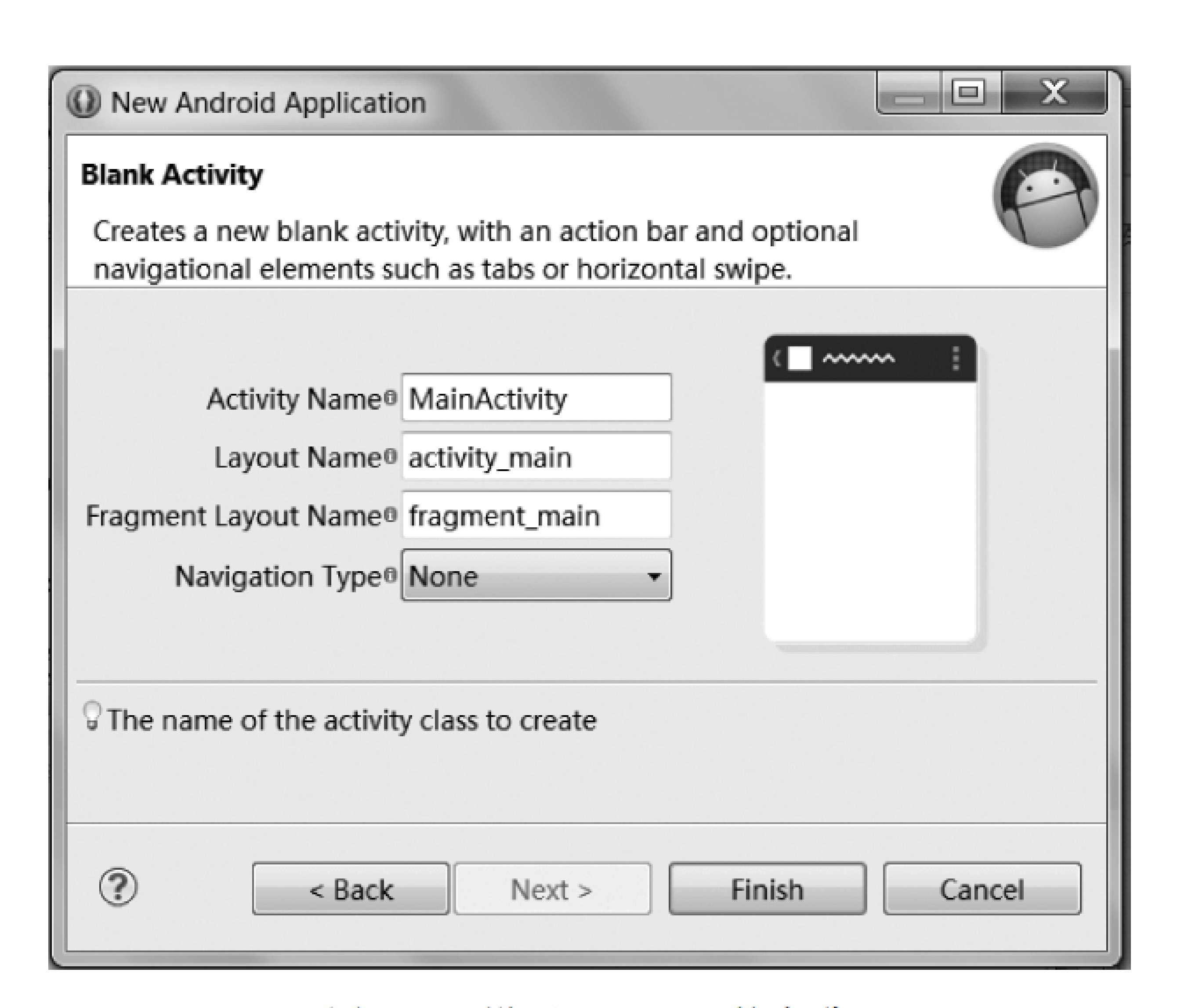


图 2.5　设置 Activity 的名称

2.1.2　运行 Android 应用程序

运行 Android 应用程序，可采用模拟器或移动设备两种方式，无论哪一种方式，都将编译生成后的二进制代码、资源文件和配置文件等打包成 APK(Android Package)文件，即 Android 的安装包，然后将这个 APK 文件发送到模拟器或移动设备上去安装运行，其过程为编译→打包→安装→运行。

1. Android 应用程序的运行

1）在模拟器上运行

选择在 Eclipse 中的 Android 项目 FirstAndroidApplication，选择菜单 Run→Run AS→Android Application，或右击该项目，在弹出的菜单中选择 Run AS→Android Application 命令，都可编译该项目，并将该项目的 APK 文件送到模拟器上安装运行，其安装运行过程的日志文件如图 2.6 所示。

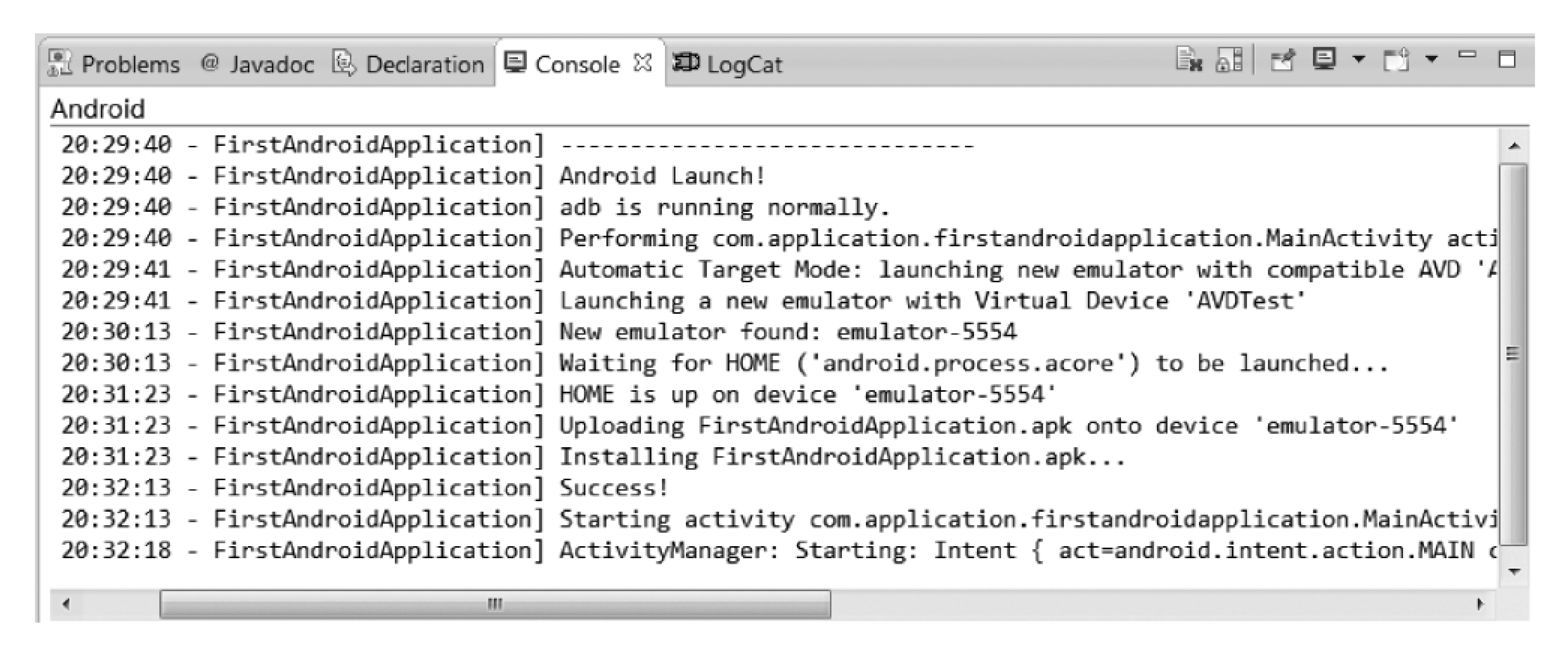


图 2.6 APK 文件送到模拟器上安装运行的日志显示

在模拟器上进入如图 2.7 所示的应用程序界面，可以看到安装到模拟器上的第一个 Android 应用程序，该程序运行结果如图 2.8 所示。

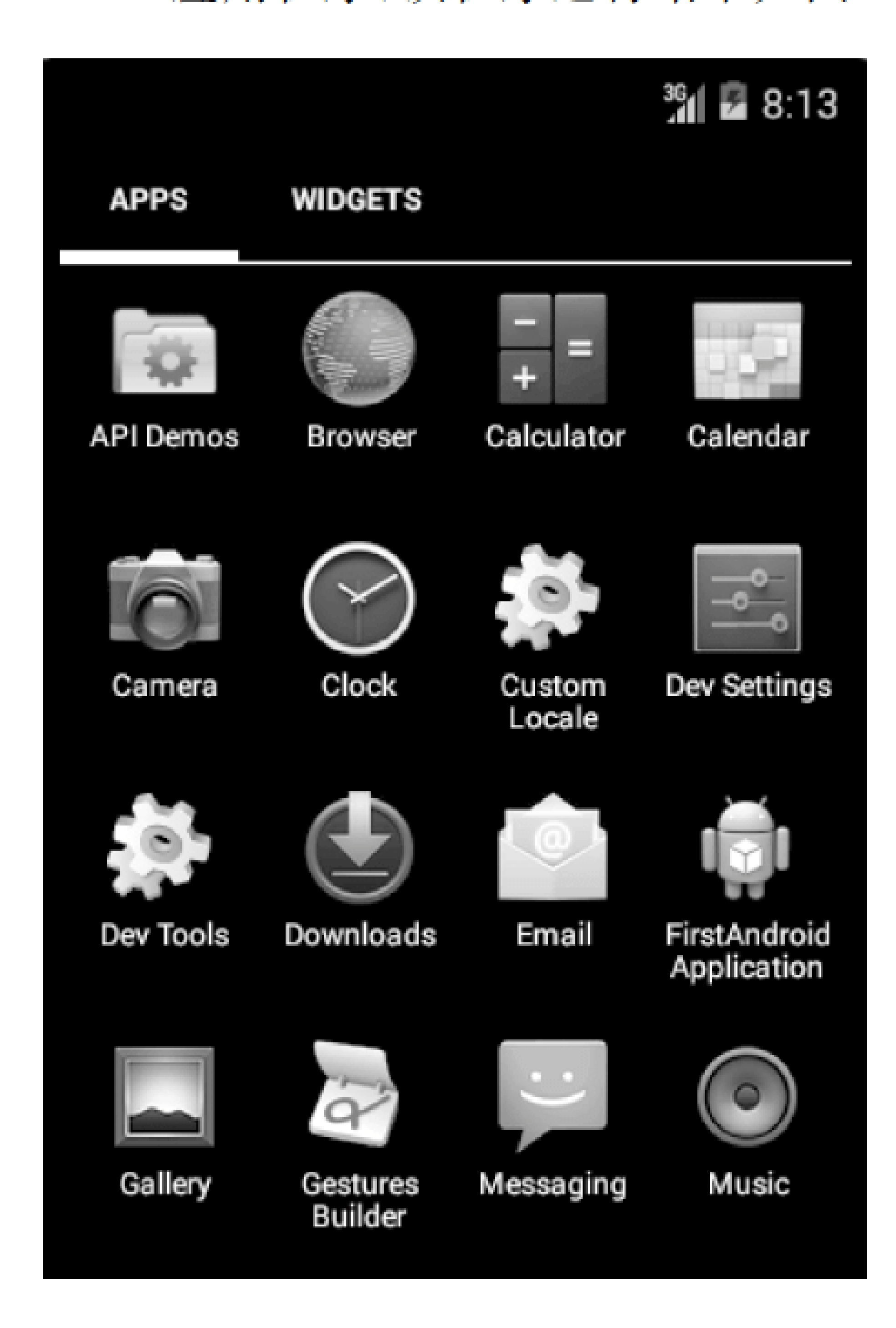




图 2.7 安装到模拟器上的第一个 Android 应用项目

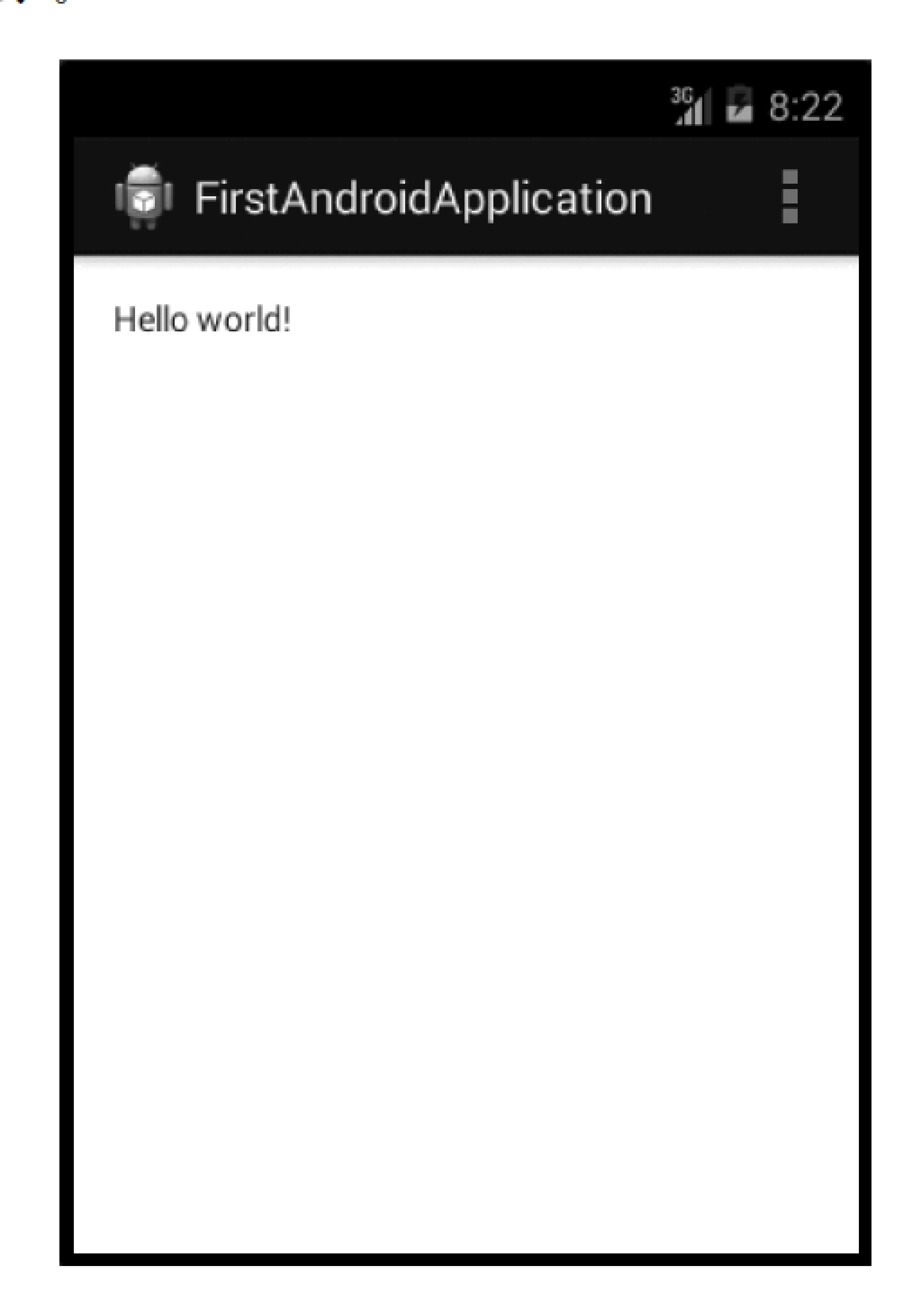


图 2.8 第一个 Android 应用项目运行结果

2）在 Android 移动设备上运行

下面以 Android 移动设备中的手机为例说明运行步骤。

(1) 下载和安装手机 USB 驱动程序。

不同手机厂商的 USB 驱动程序可到该手机厂商的官网进行下载，例如，Samsung 手机 USB 驱动程序可下载 Samsung 的 Keis 软件，网址为 http://www.samsung.com/cn/support/usefulsoftware/KIES/JSP，下载后在计算机上安装 Keis 软件。

(2) 运行手机调试模式

在计算机上运行 Keis 软件。

用 USB 连接线将 Android 手机连接到计算机上。

在手机上选择“系统设置”→“开发者选项”→“USB 调试”。

提示：APK 文件发送到移动设备上运行，比模拟器运行速度更快，效果更好。条件允许时可选用。

(3) 在手机上运行应用程序。

选择 Eclipse 中的 Android 项目 FirstAndroidApplication，右击，在弹出的菜单中选择 Run AS→Run Configurations…→Target 命令，弹出 Android Device Chooser 对话框，如图 2.9 所示，选中 samsung-gt_i8262d-26605539 选项。

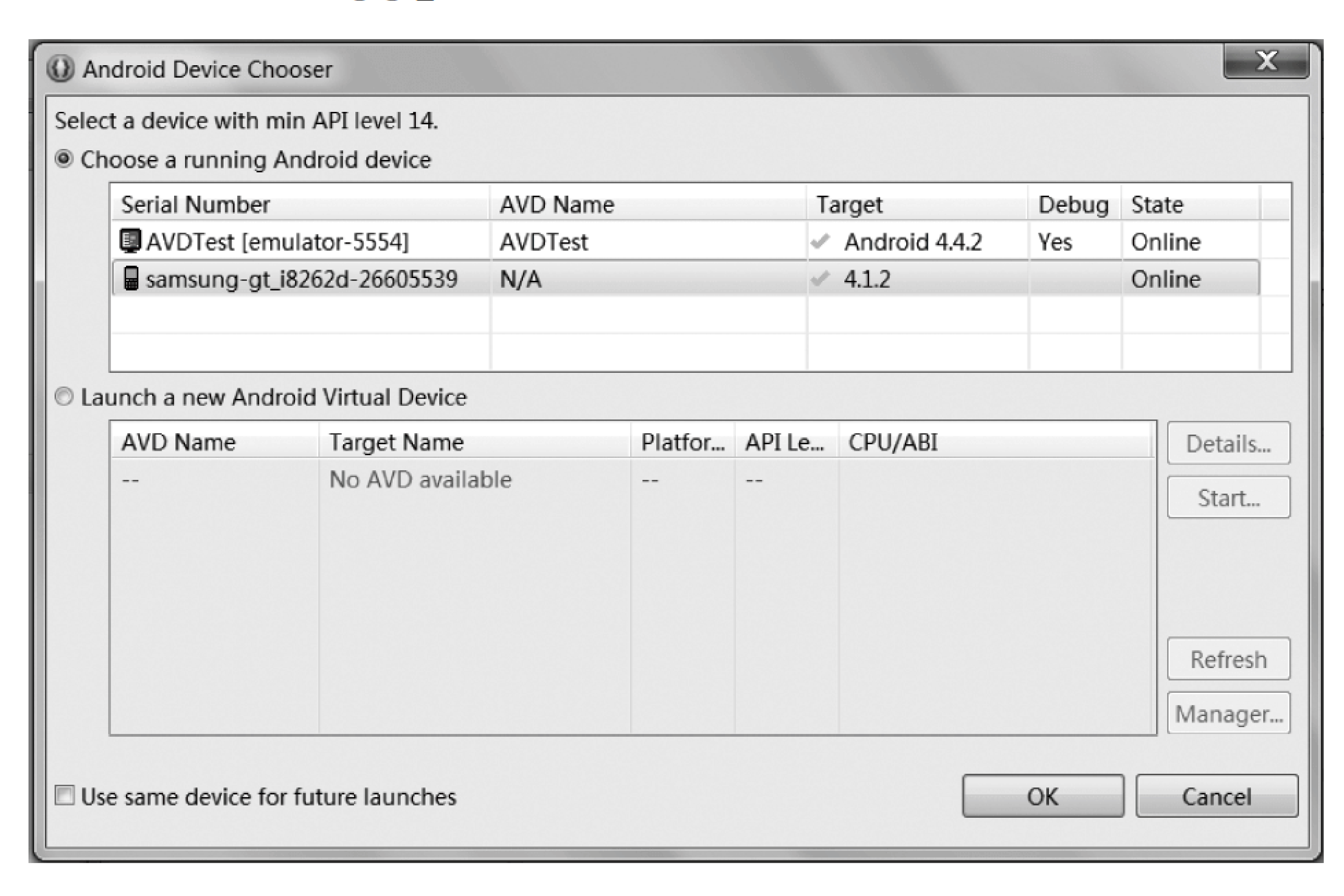


图 2.9　选择导出目标

单击 OK 按钮，即在手机上显示运行结果，其安装运行过程的日志文件如图 2.10 所示。

提示：APK(Android Package)文件指 Android 的安装包，将 Android 应用程序编译生成后的二进制代码、资源文件和配置文件等打包成 APK 文件，然后将它发送到模拟器或移动设备上去安装运行。

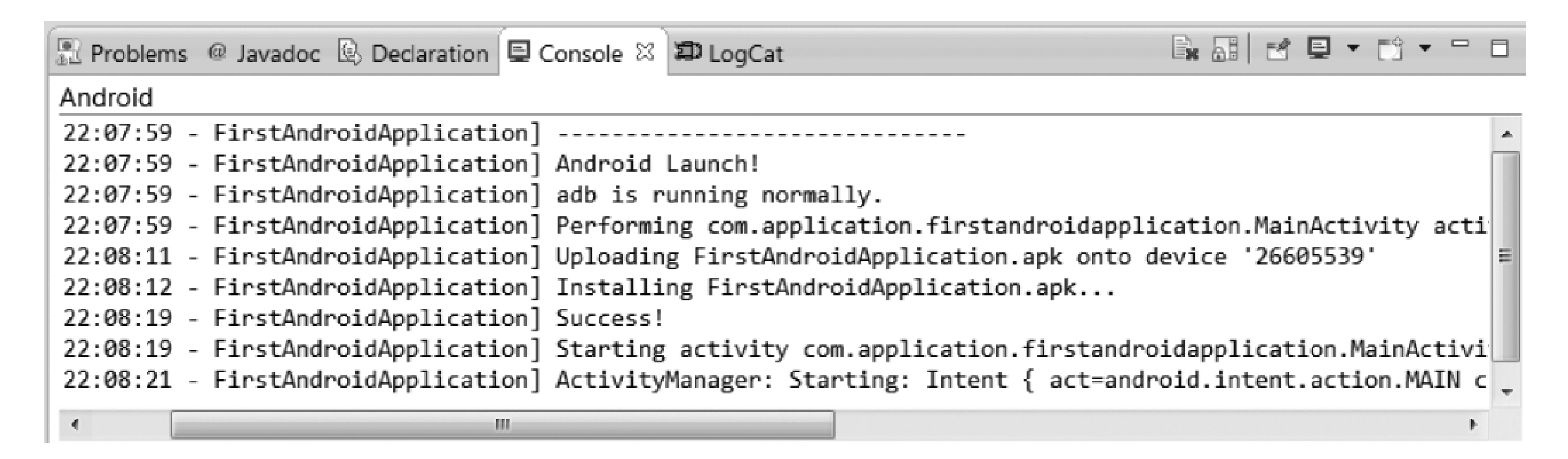


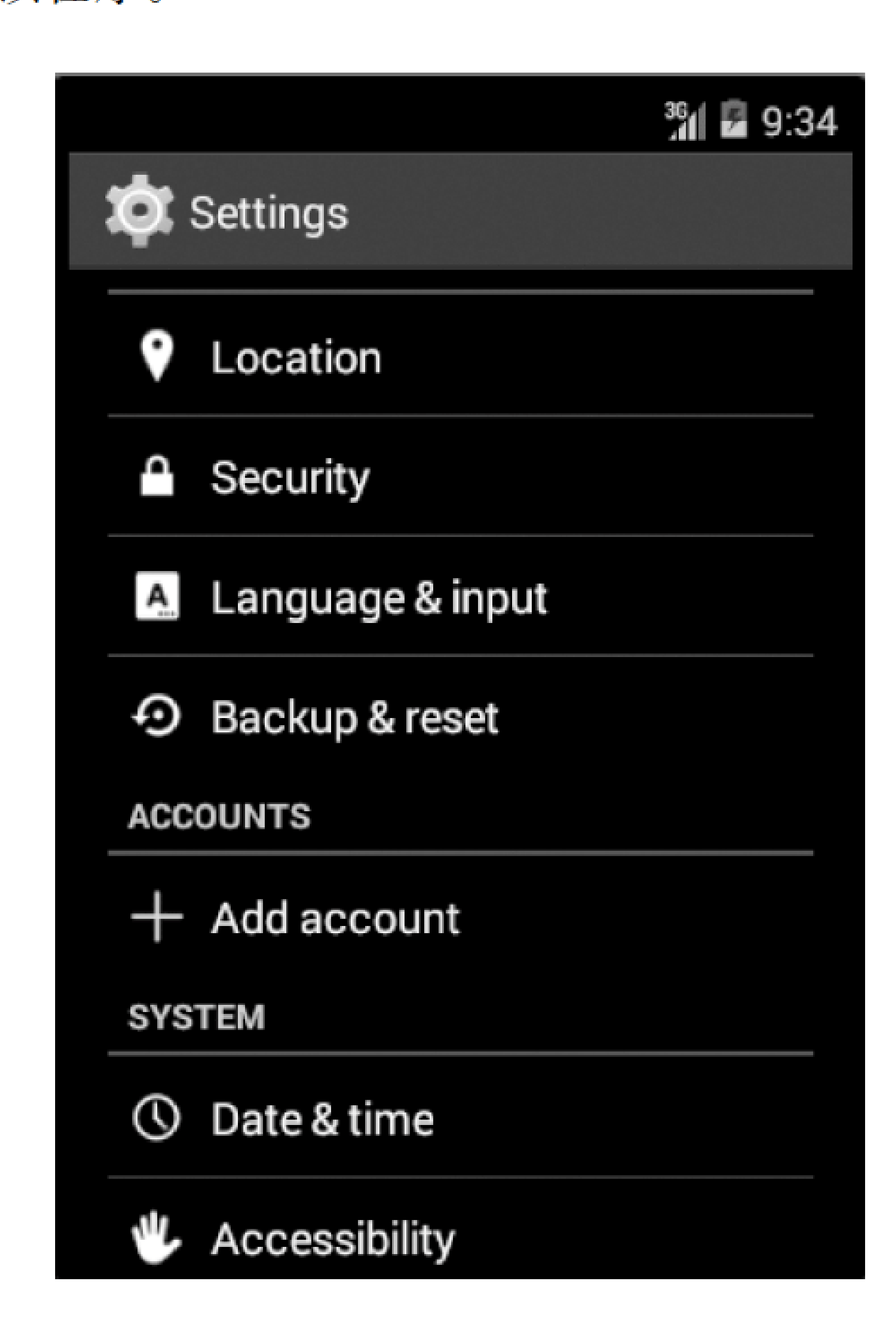
图 2.10　APK 文件送到手机上安装运行的日志显示

2. Android 应用程序的设置

单击模拟器右侧键盘面板中的 MENU 按钮，在弹出的菜单中选择 System settings 命令，出现如图 2.11 所示的设置应用程序界面，可对日期和时间、打印等进行设置。

3. Android 应用程序的卸载

单击模拟器右侧键盘面板中的按钮，在弹出的菜单中选择 Manage apps 命令，出现如图 2.12 所示的卸载应用程序界面，在应用程序列表中选择需要删除的应用程序，例如，单击 FirstAndroidApplication，打开该应用程序的应用信息界面，单击 Uninstall 按钮，即可卸载该程序。



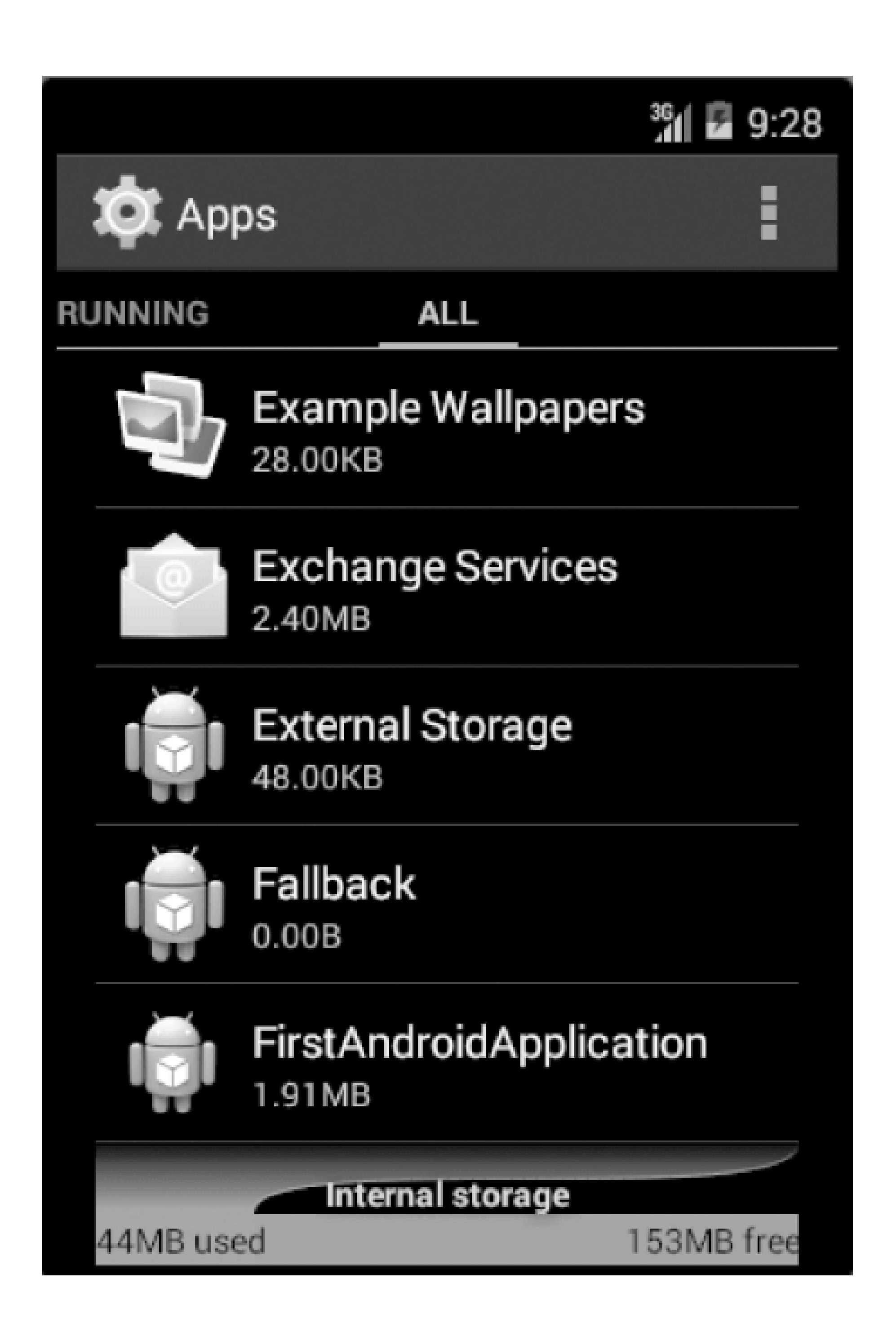




图 2.11　设置应用程序

图 2.12　卸载应用程序

2.1.3 Android 项目的导入、导出和移除

从事 Android 项目开发，时常需要将已完成的项目从 Eclipse 工作区备份到其他机器上，也常常需要借鉴别人已开发好的项目，因此，项目的导出、导入和移除在开发工作中是重要的基本操作。

1. 导出 Android 项目

选择在 Android 开发环境界面中的 Package Explorer 面板中的 Android 项目，选择菜单 File→Export→General→File System，可以将该项目文件导出到指定的文件夹中。

项目导出过程如下：

(1) 右击项目名 FirstAndroidApplication，在弹出的菜单中选择 Export…菜单项，出现 Export 对话框，选择 General→File System，单击 Next 按钮，如图 2.13 所示。

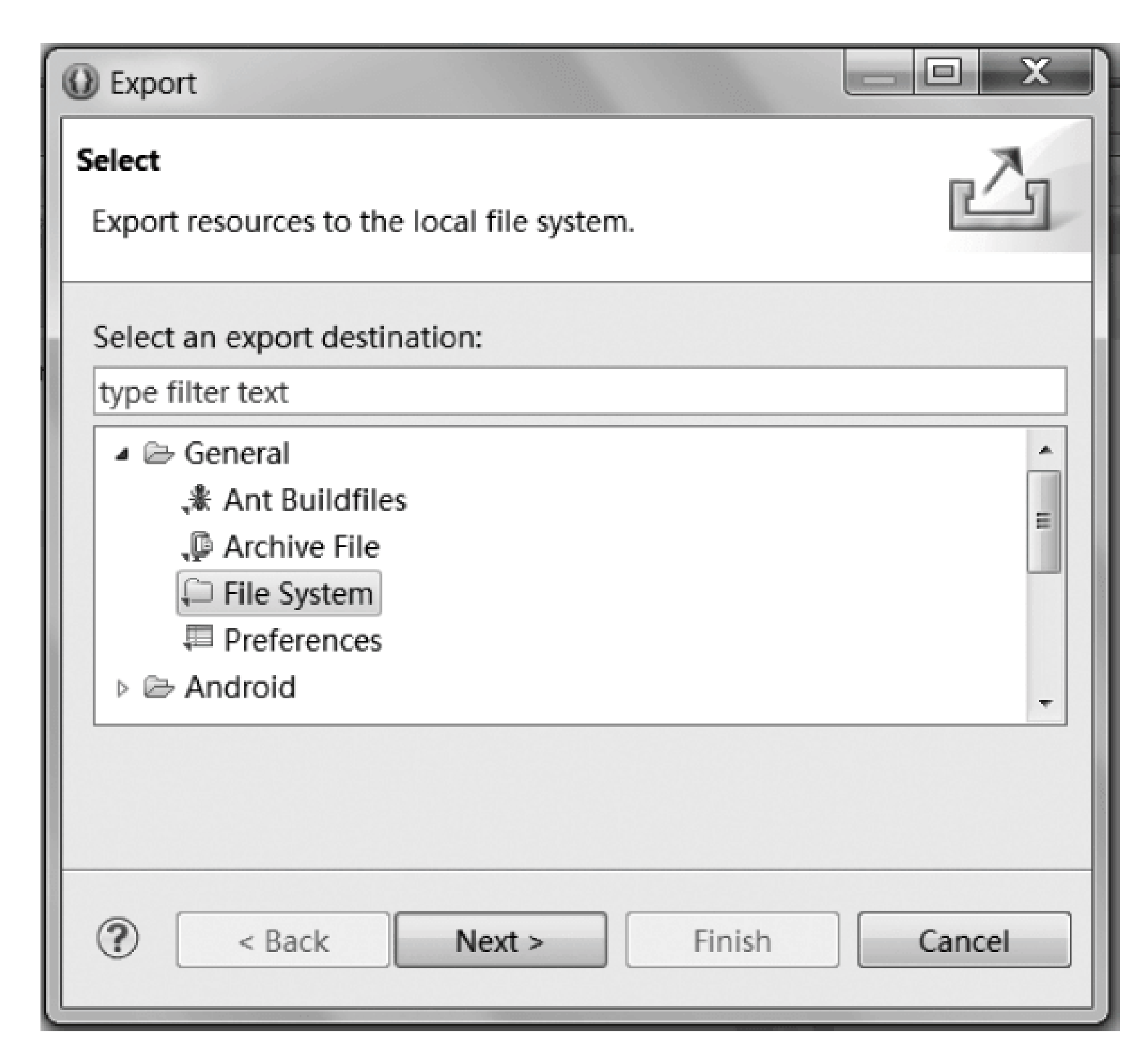


图 2.13 选择导出目标

(2) 单击 Browse…按钮，选择存盘路径，这里是 E:\SrcAndroid，如图 2.14 所示。

单击 Finish 按钮，导出完成，可在该路径下找到导出的项目。

2. 移除 Android 项目

选择在 Android 开发环境界面中的 Package Explorer 面板中的 Android 项目，右击，在弹出的菜单中选择 Delete 命令，可移除该项目。

项目移除过程如下：

(1) 右击项目名 FirstAndroidApplication，在弹出的菜单中选择 Delete 菜单项，出现 Delete Resources 对话框，单击 OK 按钮，如图 2.15 所示，此时，Eclipse 右边项目目录树中的项目 FirstAndroidApplication 已消失，表明已被移除。

注意：移除之后的项目文件仍然存在于工作区目录下，需要时可重新导入。

(2) 若要彻底删除项目，只需在图 2.15 中选择 Delete project contents on disk (cannot

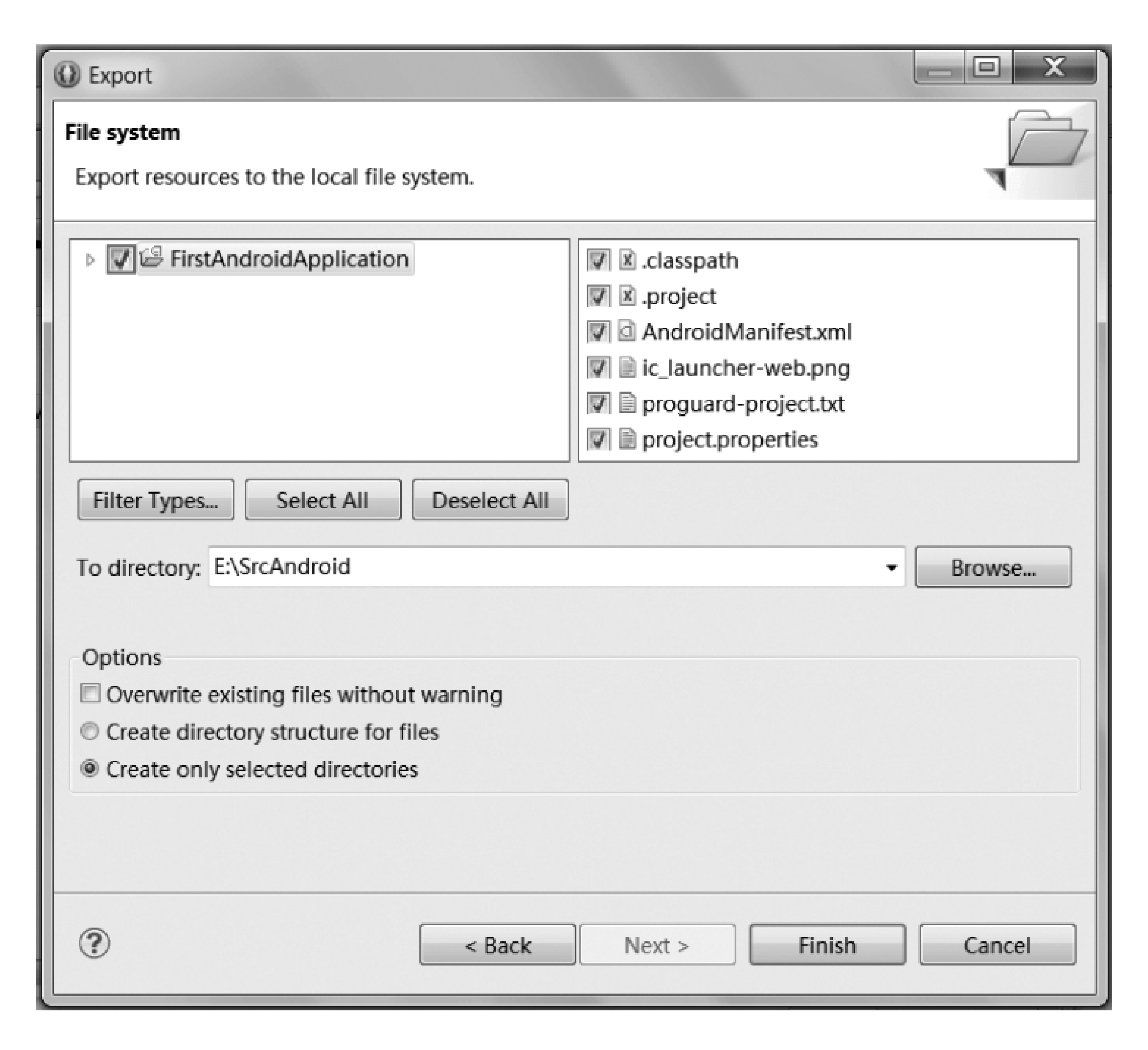


图 2.14　选择导出文件夹

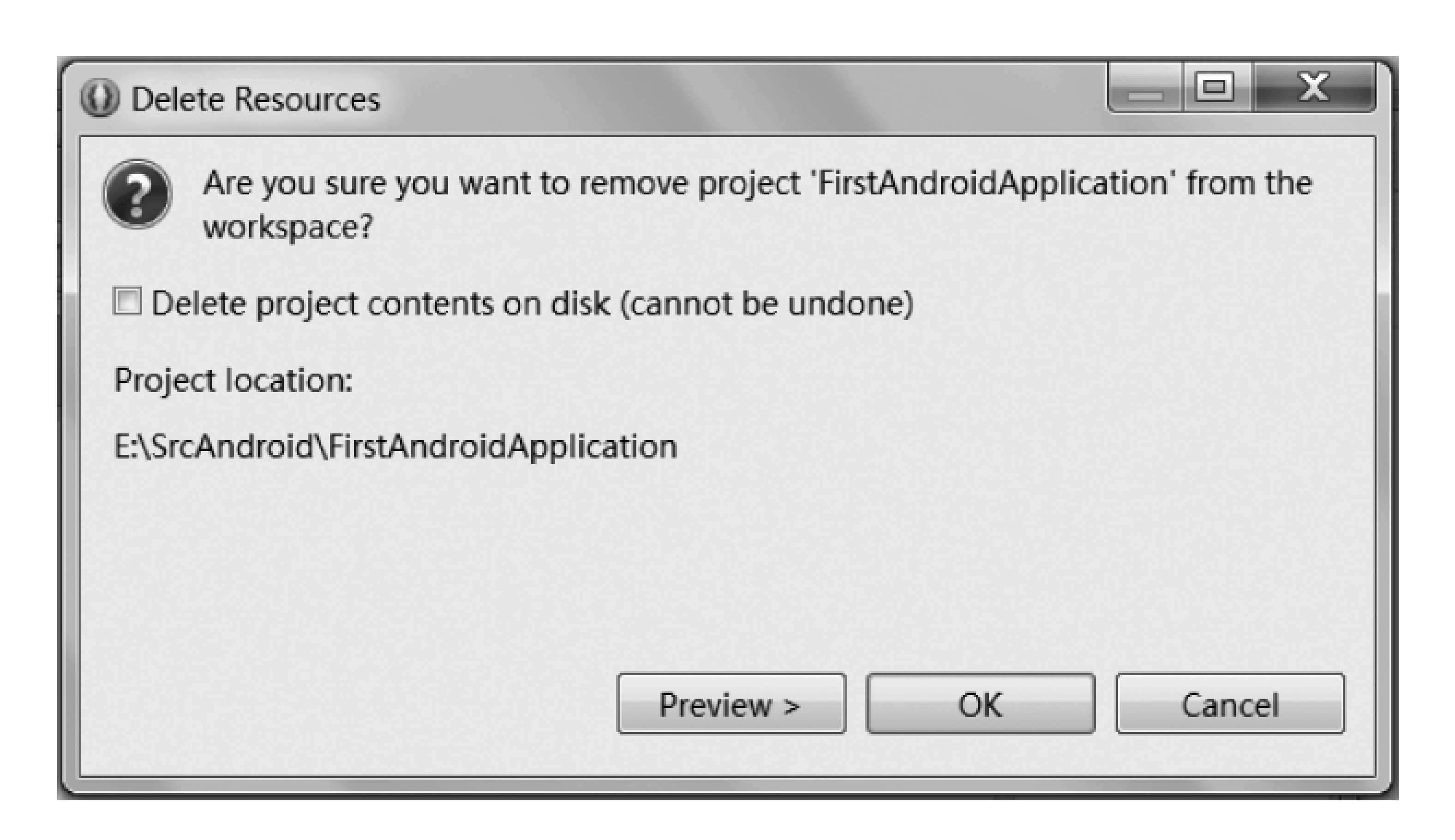


图 2.15　确认移除项目

be undone)复选框，单击 OK 按钮，即将该项目的源文件一并删除。

3. 导入 Android 项目

在 Android 开发环境界面中，选择菜单 File→Import→Android→Exiting Android Code Into Workspace，可以将一个已有的 Android 项目文件导入到当前工作空间，即导入到图 1.7 左侧的 Package Explorer 面板中。

项目导入过程如下：

(1) 在 Eclipse 主菜单选择 File→Import…，出现 Import 对话框，选择 General→Existing Android Code Into Workspace，单击 Next 按钮，如图 2.16 所示。

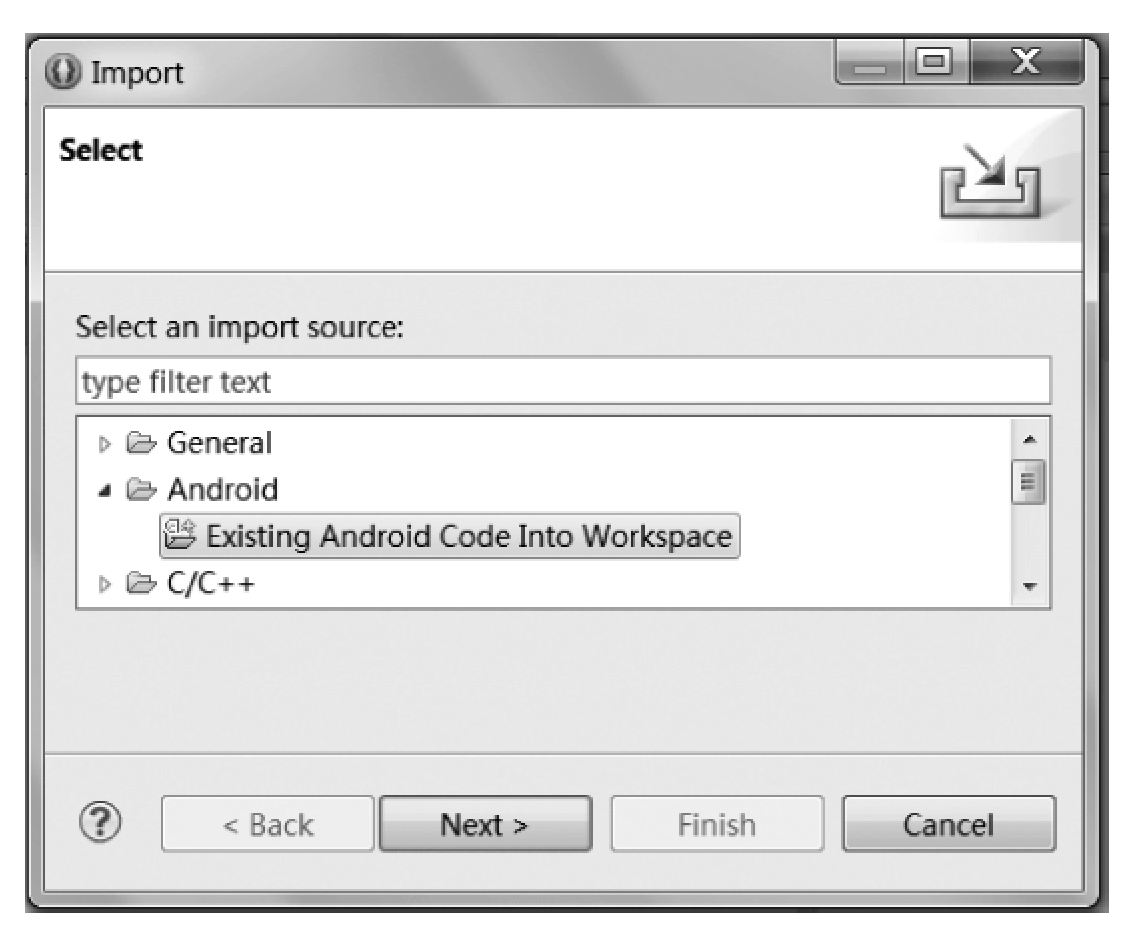


图 2.16　导入已存在的项目

(2) 出现 Import 对话框，单击 Browse…按钮，选择要导入的项目，出现"浏览文件夹"对话框，这里选择导入项目 FirstAndroidApplication，单击 Finish 按钮。

(3) 出现如图 2.17 所示的对话框，单击 Finish 按钮，完成导入工作。



图 2.17　完成导入

导入完成后，可在 Eclipse 左边项目目录树中找到导入的项目。

注意：在 Android 应用项目开发工作中，经常需要将已完成的项目从 Eclipse 工作区备份到其他机器上，也时常需要从其他机器上将项目借鉴到工作区，因此，项目的导出、导入和移除是重要的基本操作，读者应及时移除暂时不运行的项目，养成"运行一个，导入一个，运行完即移除，需要时再导入"的良好习惯。

2.2 Android 应用的目录结构

开发 Android 应用程序包含以下步骤：

(1) 创建一个 Android 应用项目。

(2) 在 XML 文档中定义用户界面。

(3) 在 Java 代码中编写业务实现。

Android 应用程序由 Java 代码和 XML 文档共同设计完成，每个 Android 应用项目都用一个目录结构来组织，例如，图 2.18 就是 FirstAndroidApplication 应用项目的目录结构。

下面对目录结构进行介绍。

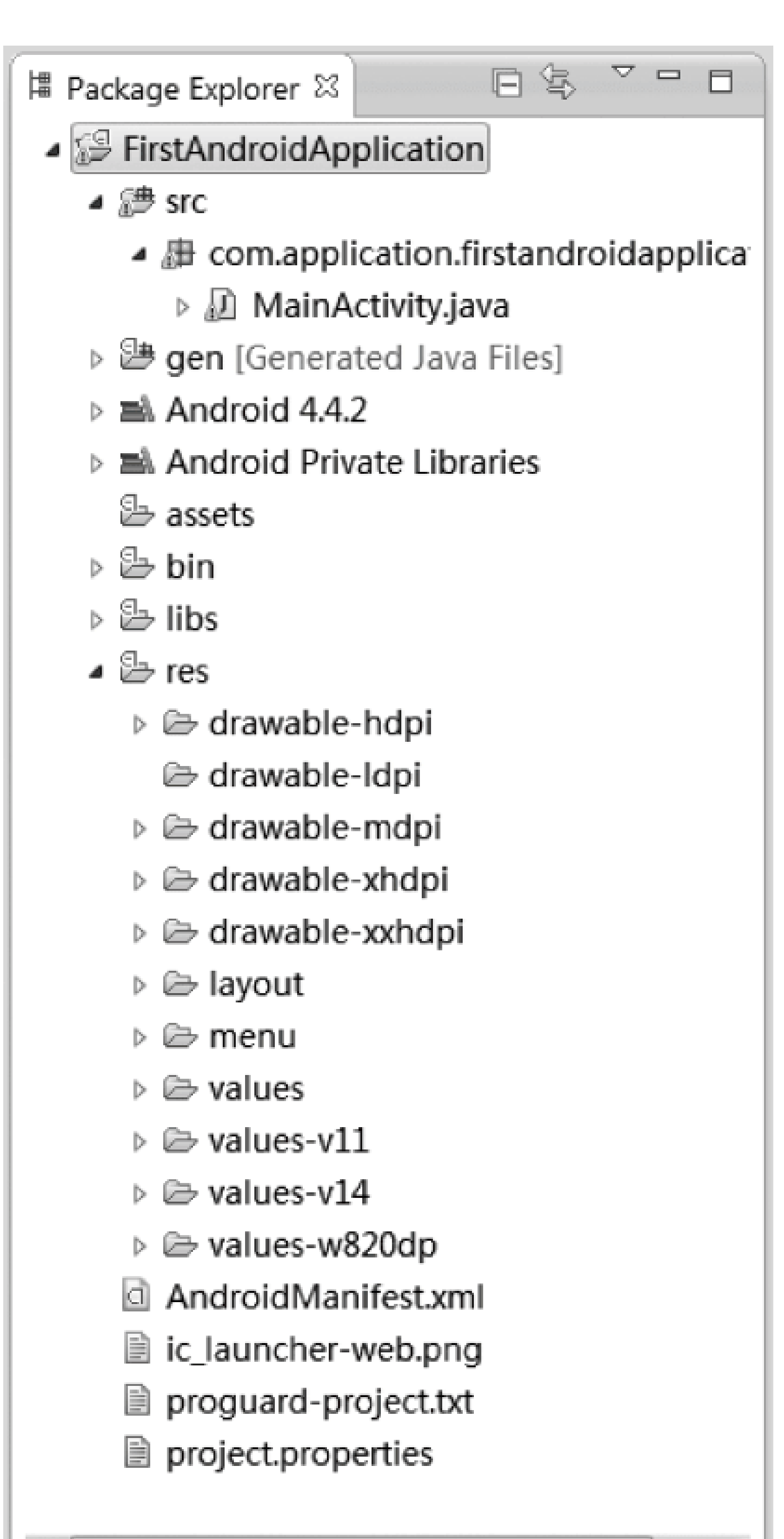


图 2.18 FirstAndroidApplication 应用项目的目录结构

1. src 目录

src 目录为源代码目录，存放 Android 应用程序中的 Java 源代码，以用户定义的包自动地进行组织。例如，在 FirstAndroidApplication 应用项目中，用户定义的包为 com.application.firstandroidapplication，那么 MainActivity.java 就在这个包内，其目录为 src/com/application/firstandroidapplication/MainActivity.java。程序员主要的工作就是编写该目录下的源代码文件。

提示：包(Package)是 Java 语言提供的一个管理名字空间的机制，是类的组织方式，每个包对应一个目录结构。

2. gen 目录

gen 目录下的文件是由 ADT 自动生成的 Java 文件，其中的 R.java 文件为项目中的各个资源在该类中创建其唯一的 ID，当项目使用这些资源时，可以通过 ID 得到该资源的引用。

注意：R.java 文件由 ADT 自动生成，用户不要去修改这个文件。

3. Android 4.4.2 目录

存放支持项目的 JAR 包。

4. assets 目录

存放与项目相关的资源文件，如音频文件、视频文件等，这个目录使用不多。

5. bin 目录

bin 目录用于存放生成的目标文件，如 Java 的二进制文件、Dalvik 虚拟机的可执行文件(.dex)、FirstAndroidApplication.apk 文件等。

6. res 目录

res 目录存放整个项目所用的全部资源文件，包括所有图形、布局和字符串资源等文

件。该目录使用很多，当存放的资源文件发生变化时，R. java 文件会依据变化自动修改。

新建一个项目，在该目录中会自动建立以下目录及其文件。

（1）drawable-ldpi、drawable-mdpi、drawable-hdpi、drawable-xhdpi、drawable-xxhdpi 等 5 个目录。

分别存储低、中、高、超高分辨率的图形文件，文件类型有. png、. 9. png、. jpg 等格式。

（2）layout 目录：存放应用程序的布局文件，文件类型为 XML 格式。Android 在 XML 文件中使用 XML 元素来设定屏幕的布局。

（3）values 目录：存放所有 XML 格式的资源描述文件，例如字符串（strings. xml）、颜色（colors. xml）、样式（styles. xml）、尺寸（dimens. xml）和数组（arrays. xml）等。

7. AndroidManifest. xml 文件

Android 项目配置文件，它是项目的系统控制文件，用于控制应用的名称、图标、访问权限等整体属性，每个项目必需的文件，在程序中定义组件需要在这里注册，也可给应用程序在这个文件中添加权限声明，这个文件经常用到。

8. proguard-project. txt 文件

该文件是混淆代码的脚本配置文件，用于保护源代码。

9. project. properties 文件

指定编译程序时使用的 SDK 版本。

2.3 Android 应用程序分析

Android 应用开发建立在应用程序框架之上，开发简捷、方便，用户界面由 XML 担任，业务逻辑由 Java 程序完成，以实现界面设计和程序逻辑的分离，具有低耦合性和高重用性，可以让程序设计人员集中精力开发业务逻辑，界面设计人员集中精力进行界面设计，这是符合 MVC 设计思想的，从而达到提高开发效率、缩减开发时间，降低开发成本的目的。

提示：MVC 是一种设计模式，将一个应用划分为以下层面：M（Model，模型层）：封装应用程序的业务逻辑和数据结构，负责数据的存取；V（View，视图层）：它是模型层的外在表现，负责界面的显示；C（Controller，控制器层）：将模型层和视图层联系到一起，确保模型层和视图层的同步，负责将数据写到模型层中并调用视图。

下面对第一个 Android 应用项目 FirstAndroidApplication 的程序进行解析。

我们关心第一个 Android 应用项目运行结果（如图 2.8 所示）：界面和“Hello world!”从何而来？需要从访问源代码目录做起。

2.3.1 源代码文件

在 Android 应用项目目录 src 中存放的应用程序是 Java 源代码文件，自动地组织在用户声明的包内。

1. src 目录

在应用项目 FirstAndroidApplication 的目录 src/com. application. firstandroidapplication MainActivity. java 文件中，重写 onCreate() 方法，调用父类的 onCreate() 方法，调用

setContentView()方法,加载 activity_main. xml 布局文件。

在该文件中编辑代码如下:

```
1  package com.application.firstandroidapplication;
2
3  import android.app.Activity;
4  import android.app.ActionBar;
5  import android.app.Fragment;
6  import android.os.Bundle;
7  import android.view.LayoutInflater;
8  import android.view.Menu;
9  import android.view.MenuItem;
10 import android.view.View;
11 import android.view.ViewGroup;
12 import android.os.Build;
13
14 public class MainActivity extends Activity {
15
16     //重写 onCreate()方法
17     @Override
18     protected void onCreate(Bundle savedInstanceState) {
19         //调用父类的 onCreate()方法
20         super.onCreate(savedInstanceState);
21         //调用 setContentView()方法,显示 activity_main.xml 文件中定义的屏幕布局
22        setContentView(R.layout.activity_main);
23 …
24     }
25…
26 }
```

(1) 第 1 行至第 12 行,声明包 package com. application. firstandroidapplication,引入有关类 android. app. Activity 等。

提示:Activity 类是 Android 系统提供的一个活动基类,项目中所有活动需要继承它才能具有活动特性。

(2) 第 14 行定义类 MainActivity 的开始,extends 关键字表示 MainActivity 类继承自 Activity 类,在第 3 行引入。

提示:继承(Inheritance)可以实现代码的复用,被继承的类称为父类、基类或超类,由继承而得的类称为子类或导出类。子类继承父类的成员变量和成员方法,可以修改父类的成员变量或重写父类的方法,还可以添加新的成员变量和成员方法。

(3) 第 16 行是 Java 源代码的单行注释。

(4) 第 17 行@Override 表示重写下一行紧跟的方法 onCreate(),MainActivity 类继承自 Activity 类,Activity 类已定义 onCreate()方法,重写 onCreate()方法即重新定义 MainActivity 类自己的 onCreate()方法,当程序运行时,就会执行 MainActivity 类自己的

onCreate()方法里的代码。

提示：重写(Override)是在子类中重新定义自己的方法，但方法名称、参数个数与类型和父类完全相同。

(5) 第 17 行至第 24 行，重新定义 onCreate() 方法，传入一个 Bundle 类型参数 savedInstanceState。

(6) 第 20 行，通过 super. onCreate() 方法调用 MainActivity 类的父类 Activity 的 onCreate()方法，Bundle 类型参数 savedInstanceState 用于保存当前 Activity 的信息。

(7) 第 22 行，调用了 setContentView()方法，给当前活动引入一个布局，即显示 layout 目录中 activity_main. xml 文件中定义的屏幕布局，界面和“Hello world!”应该在这个布局中，让我们打开这个文件看看。

2. Java 文件注释

注释是对 Java 程序功能的解释或说明，其目的是提高程序的可读性，为阅读和理解程序提供方便，有利于程序开发人员进行交流和合作，从而提高开发效率，注释内容都会被编译器忽略。

对 Java 源代码进行注释，有以下 3 种类型：

(1) 多行注释，以/*开始，以*/结束的一行或多行文字。

例如：

```
/* 重写 onActivityResult()方法,通过判断请求码值和返回码值,来确定是否正确获得回传数据,
   如果是正确的,则取出回传数据并显示在标题栏中 */
```

(2) 单行注释，以双斜杠//开头，在该代码行的末尾结束。

例如：

```
//创建一个 Bundle 对象,对象名为 bundle
```

(3) 文档注释，以/**开始，以*/结束的多行。

文档注释是 Java 特有的，主要用来生成类定义的 API 文档。

2.3.2 资源文件

在 Android 应用项目目录 res 中存放整个项目所用的全部资源文件，下面介绍其中的 layout 目录和 values 目录。

1. layout 目录

存放应用程序的布局文件。

在应用项目 FirstAndroidApplication 的目录 res/layout 下，有 activity_main. xml 文件和 fragment_main. xml 文件。

(1) activity_main. xml 文件代码如下：

```
<FrameLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    xmlns:tools = "http://schemas.android.com/tools"
    android:id = "@+id/container"
    android:layout_width = "match_parent"
```

```
5        android:layout_height = "match_parent"
6        tools:context = "com.application.firstandroidapplication.MainActivity"
7        tools:ignore = "MergeRootFrame" />
```

（2）fragment_main.xml 文件代码如下：

```
1  < RelativeLayout xmlns:android = "http://schemas.android.com/apk/res/android"
2      xmlns:tools = "http://schemas.android.com/tools"
3      android:layout_width = "match_parent"
4      android:layout_height = "match_parent"
5      android:paddingBottom = "@dimen/activity_vertical_margin"
6      android:paddingLeft = "@dimen/activity_horizontal_margin"
7      android:paddingRight = "@dimen/activity_horizontal_margin"
8      android:paddingTop = "@dimen/activity_vertical_margin"
9      tools:context = "com.application.firstandroidapplication.MainActivity$PlaceholderFragment" >
10     <! -- 设置一个 TextView 控件 -- >
11     < TextView
12         android:layout_width = "wrap_content"
13         android:layout_height = "wrap_content"
14         android:text = "@string/hello_world" />
15 </RelativeLayout >
```

① 在上述两个文件的第 1 行标签中，都有属性 xmlns:android＝“http://schemas.android.com/apk/res/android”，这是 XML 命名空间的声明，用于告诉 Android 工具，涉及的公共属性已定义在命名空间，每一个 Android 布局文件的最外层标签必须有这个属性。

② 在 fragment_main.xml 文件中，有一个文本控件 TextView，该控件由 Android 提供，用于在布局文件中显示文字，在 TextView 中，代码行 android:text＝“@string/hello_world”提示我们在 string.xml 文件中的 hello_world 中会有结果，运行结果快要找到了！

③ 第 10 行是对 XML 文件中的代码进行注释。

2. values 目录

存放所有 XML 格式的资源描述文件，例如字符串(strings.xml)、颜色(colors.xml)、样式(styles.xml)、尺寸(dimens.xml)和数组(arrays.xml)等。

在应用项目 FirstAndroidApplication 的目录 res/values 下的 strings.xml 文件，其代码如下：

```
1  <?xml version = "1.0" encoding = "utf - 8"?>
2  < resources >
3      < string name = "app_name"> FirstAndroidApplication </string >
4      < string name = "hello_world"> Hello world! </string >
5      < string name = "action_settings"> Settings </string >
6  </resources >
```

（1）第 1 行声明 XML 文件的版本号和编码方式，下一行是定义资源的标签。

（2）第 3 行至第 5 行定义了 3 个字符串常量，字符串的名称分别是 app_name、hello_world 和 action_settings，相应内容分别是 FirstAndroidApplication、Hello world！和

Settings。

Hello world! 值对应 hello_world 键,在布局文件 activity_main. xml 和 fragment_main. xml 中,通过引用 hello_world 键,找到相应值,到此,我们终于找到第一个 Android 应用项目运行结果的来源。

提示:Android 应用开发建立在应用程序框架之上,用户界面 V 由 XML 担任,业务逻辑 M 由 Java 程序完成,以实现界面设计和程序逻辑的分离,具有低耦合性和高重用性、符合 MVC 设计思想。

3. XML 文件注释

对 XML 文件中的代码进行注释,以<! --开始,以-->结束的一行或多行文字。

例如:

```
<! -- 设置一个 ImageView 控件 -->
```

2.3.3 资源索引文件

gen 目录下的 R. java 文件由 Android 自动生成,用户不要修改,该文件为项目中的各个资源在该类中创建其唯一的 ID,当项目使用这些资源时,可以通过 ID 得到该资源的引用。

在项目 FirstAndroidApplication 的 gen 目录下的 R. java 文件的代码如下:

```
package com.application.firstandroidapplication;

public final class R {
    public static final class attr {
    }
    public static final class dimen {
        public static final int activity_horizontal_margin = 0x7f040000;
        public static final int activity_vertical_margin = 0x7f040001;
    }
    public static final class drawable {
        public static final int ic_launcher = 0x7f020000;
    }
   public static final class id {
       public static final int action_settings = 0x7f080001;
       public static final int container = 0x7f080000;
    }
    public static final class layout {
        public static final int activity_main = 0x7f030000;
        public static final int fragment_main = 0x7f030001;
    }
    public static final class menu {
        public static final int main = 0x7f070000;
    }
```

```
24    public static final class string {
25        public static final int action_settings = 0x7f050002;
26        public static final int app_name = 0x7f050000;
27        public static final int hello_world = 0x7f050001;
28    }
29    public static final class style {
30        public static final int AppBaseTheme = 0x7f060000;
31        public static final int AppTheme = 0x7f060001;
32    }
33 }
```

（1）第 1 行声明 R.java 文件所在的包为 com.application.firstandroidapplication。

（2）在以下有关行中，分别为使用维度的资源 dimen 创建 ID、为使用图片的资源 drawable 创建 ID，为使用屏幕布局的资源 layout 创建 ID，为使用字符串常量的资源 string 创建 ID，为使用样式的资源 style 创建 ID。

2.3.4 项目配置文件

Android 项目配置文件 AndroidManifest.xml 存放在项目根目录下，该文件为项目的系统控制文件，在程序中定义组件需要在这里注册，是每个 Android 项目必需的文件。

在项目 FirstAndroidApplication 根目录下的 AndroidManifest.xml 文件的代码如下：

```
<?xml version = "1.0" encoding = "utf - 8"?>
< manifest xmlns:android = "http://schemas.android.com/apk/res/android"
    package = "com.application.firstandroidapplication"
    android:versionCode = "1"
    android:versionName = "1.0" >
    < uses - sdk
        android:minSdkVersion = "14"
        android:targetSdkVersion = "19" />
    < application
        android:allowBackup = "true"
        android:icon = "@drawable/ic_launcher"
        android:label = "@string/app_name"
        android:theme = "@style/AppTheme" >
        < activity
            android:name = "com.application.firstandroidapplication.MainActivity"
            android:label = "@string/app_name" >
            < intent - filter >
                < action android:name = "android.intent.action.MAIN" />
                < category android:name = "android.intent.category.LAUNCHER" />
            </intent - filter >
        </activity >
    </application >
</manifest >
```

(1) 第 1 行声明 XML 文件的版本号和编码方式,第 2 行声明 XML 的命名空间。

(2) 第 3 行声明应用程序的包名为 com. application. firstandroidapplication。

(3) 第 7 行和第 8 行分别声明 SDK 最低版本为 14,目标版本为 19。

(4) 第 11 行声明应用程序的图标使用 drawable 目录下的 ic_launcher. png 文件。

(5) 第 15 行声明应用程序第一个运行的 Java 代码是 src 目录下包中的 MainActivity. java 文件。

(6) 第 16 行声明屏幕上的标题显示的内容是 string. xml 中常量 app_name 定义的字符串。

2.4 Android 应用的调试

调试是 Android 应用开发的重要工作,本节介绍 Android 调试工具:Java 调试器 Debug、图形化调试工具 DDMS 和获取日志信息调试工具 LogCat。

2.4.1 Java 调试器 Debug

Eclipse 提供了一个内置的 Java 调试器 Debug,提供了标准的调试功能,包括设置断点、查看变量、单步调试、挂起和恢复线程等。

在 Eclipse 主界面中,选择菜单 File→Window→Open Perspective→Debug,进入 Debug 透视图,如图 2.19 所示。此时,在 Eclipse 窗口右上角出现 Debug 按钮,单击该按钮左侧的 Java 按钮,即切换到 Java 透视图,Debug 透视图与 Java 透视图可通过单击相应按钮相互切换。

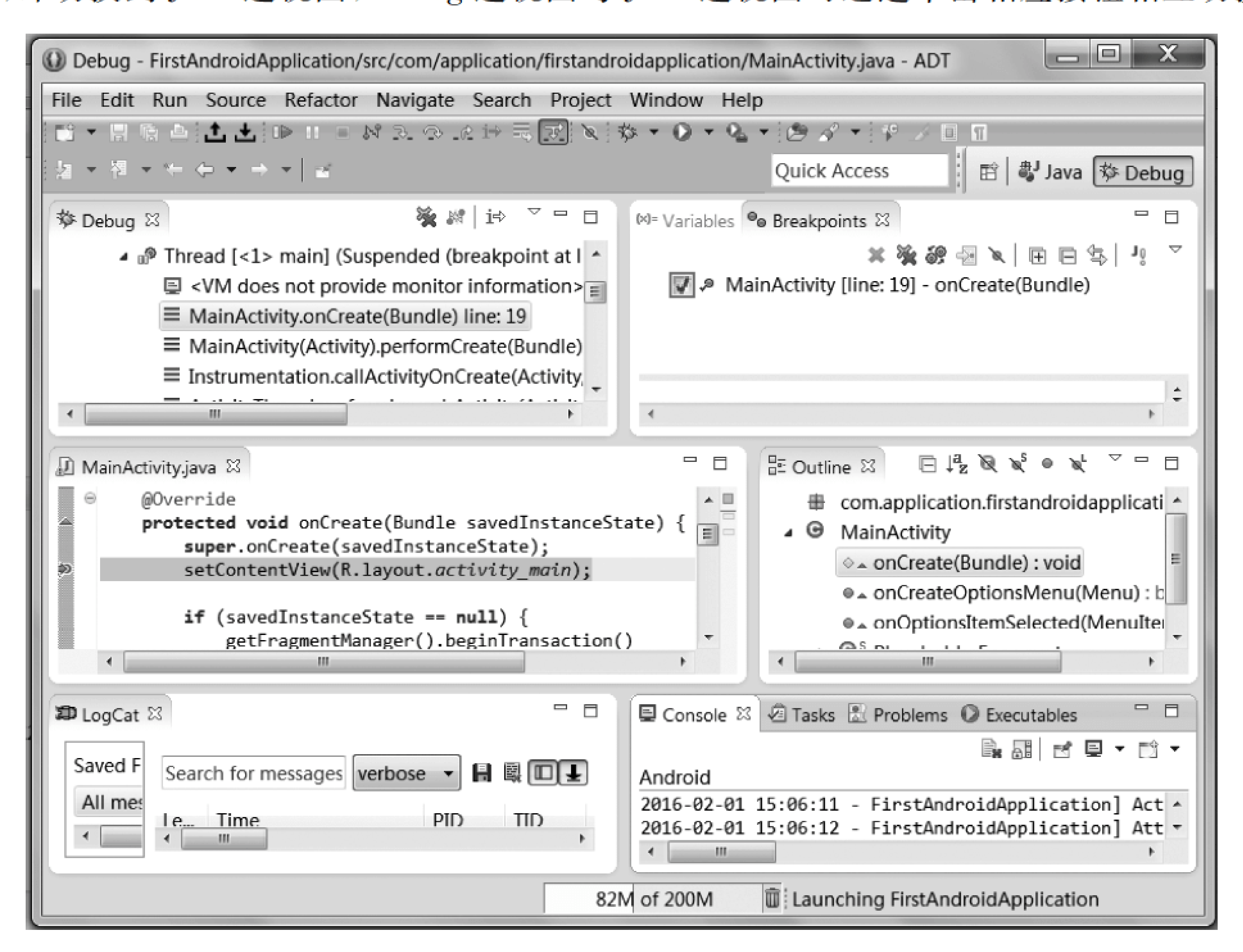


图 2.19 Debug 透视图

右击 Debug 按钮，在弹出的菜单中选择 Close 命令，即关闭 Debug 透视图。

1. Debug 透视图

Debug 透视图用于在工作台中管理程序的调试和运行，可以显示每个调试目标中挂起的线程堆栈框架。

2. Debug 面板

Debug 透视图中的 Debug 面板用于管理与程序调试相关的功能，面板中的视图呈树状结构，每一个线程对应一个树节点，在图 2.20 中显示的是暂挂线程的调试堆栈结构。

3. 设置断点

设置断点是常用的调试方法，在 Java 透视图的应用项目中，双击需要设置断点的 Java 源代码文件，该文件的代码在中部编辑器打开，将鼠标放在代码行左侧的标记栏上，双击即可设置断点。例如，在应用项目 FirstAndroidApplication 的 Java 源代码文件 MainActivity.java 中，设置了两个断点，已在 Debug 透视图中的 Breakpoints 面板中列出，如图 2.21 所示。

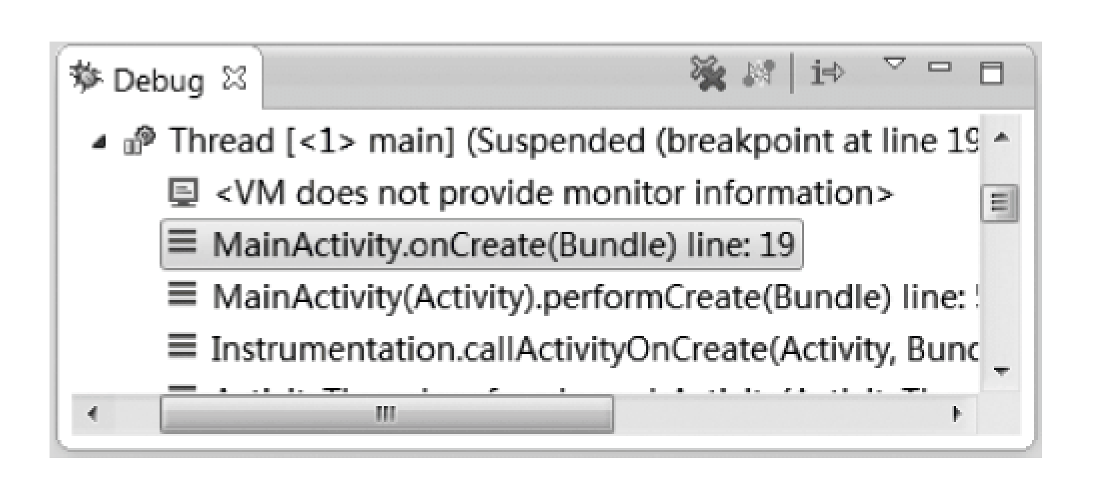


图 2.20　Debug 面板

图 2.21　BreakPoints 面板

注意：设置断点时不要将多条语句放在一行上，不能为同一行上的多条语句设置断点。

4. 查看变量

断点时的变量值，可以使用 Variables 面板查看，如图 2.22 所示。

5. 单步调试

在调试过程中，可以使用工具栏中的单步调试按钮：Step Into 按钮、Step Over 按钮、Step Return 按钮进行单步调试，如图 2.23 所示。

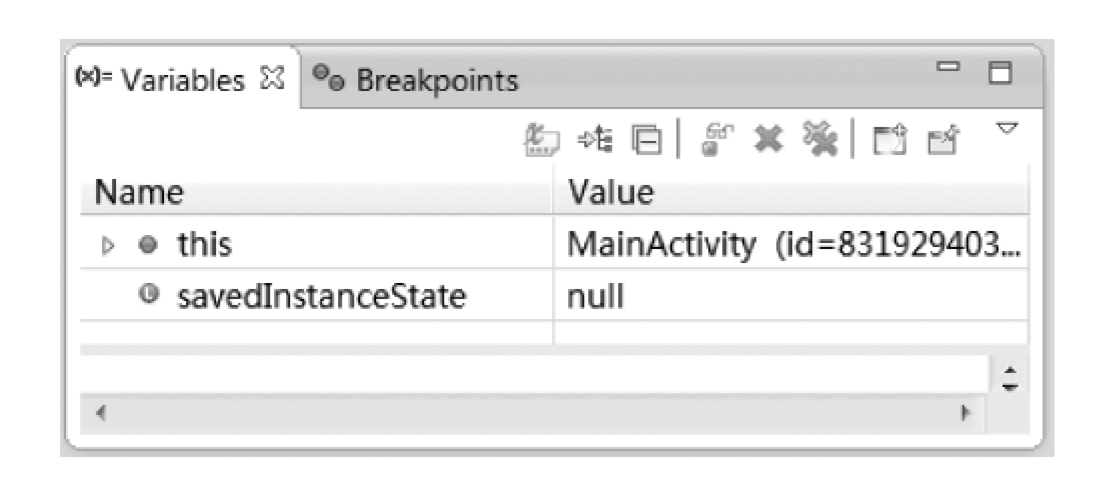


图 2.22　Variables 面板



图 2.23　工具栏中的单步调试按钮

(1) Step Into 按钮(F5)。

在单步执行过程中，遇到子函数即进入，并继续单步执行。

(2) Step Over 按钮(F6)。

在单步执行过程中，遇到子函数不进入，将整个子函数执行完再停止。

(3) Step Return 按钮(F7)。

单步执行到子函数内时,单击 Step Return 按钮则执行完子函数其余部分,并返回上一层函数。

2.4.2 图形化调试工具 DDMS

DDMS(Dalvik Debug Monitor Service)是一个功能强大的图形化调试工具,用于监控正在运行的线程以及堆信息、LogCat 日志信息、广播状态信息、模拟电话呼叫、模拟短信收发、虚拟地理位置等。

在 Eclipse 主界面中,选择菜单 File→Window→Open Perspective→DDMS,进入 DDMS 透视图,如图 2.24 所示。此时,在 Eclipse 窗口右上角出现 DDMS 按钮,单击该按钮左侧的 Java 按钮,即切换到 Java 透视图,DDMS 透视图与 Java 透视图可通过单击相应按钮相互切换。

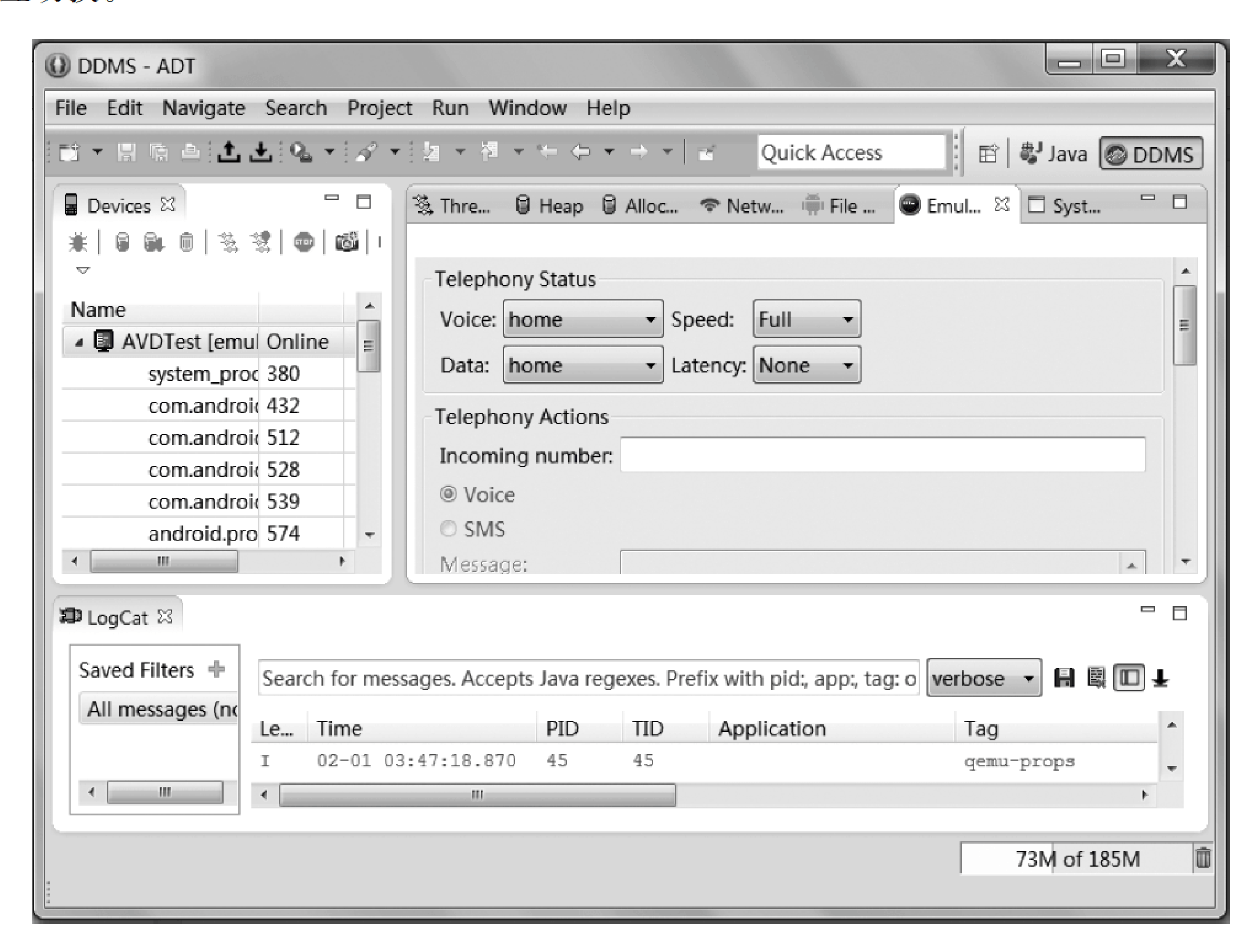


图 2.24 DDMS 透视图

右击 DDMS 按钮,在弹出的菜单中选择 Close 命令,即关闭 DDMS 透视图。

1. Devices 面板

DDMS 透视图左上部是 Devices 面板,这个设备面板列出当前所有运行的终端设备,包括模拟器和手机,并列出各个终端设备的所有进程信息。

2. DDMS 输出面板

DDMS 右上部有几个输出面板,用于获取调试中的各种输出信息。

(1) Threads 面板：线程跟踪面板，用于查看指定进程内正在执行的线程状态。为显示输出信息，需要单击 Devices 面板上部 Update Threads 按钮，并在设备面板选中需要查看的进程。

(2) Heap 面板：内存跟踪面板，用于查看指定进程内堆内存的分配和回收信息。为显示输出信息，需要单击 Devices 面板上部 Update Heap 按钮，并在设备面板选中需要查看的进程。

(3) Emulator Control 面板：模拟器控制面板，用于模拟器模拟拨打电话、发送短信，还可虚拟模拟器的位置信息。

3. LogCat 输出面板

LogCat 输出面板位于 DDMS 透视图下部，用于输出日志信息，具体使用方法参见 2.4.3 节。

2.4.3 获取日志信息调试工具 LogCat

LogCat 是 Android 系统提供的一个调试工具，用于获取系统日志信息，例如 Dalvik 虚拟机产生的信息、进程信息、ActivityManager 信息、PackagerManager 信息、Homeloader 信息、WindowsManager 信息、Android 运行时信息等。

在 Eclipse 主界面中，选择菜单 File→Window→Show View→Other…，弹出一个 Show View 对话框，选择 Android→LogCat，单击 OK 按钮，在 Eclipse 主界面下部出现 LogCat 视图，如图 2.25 所示。

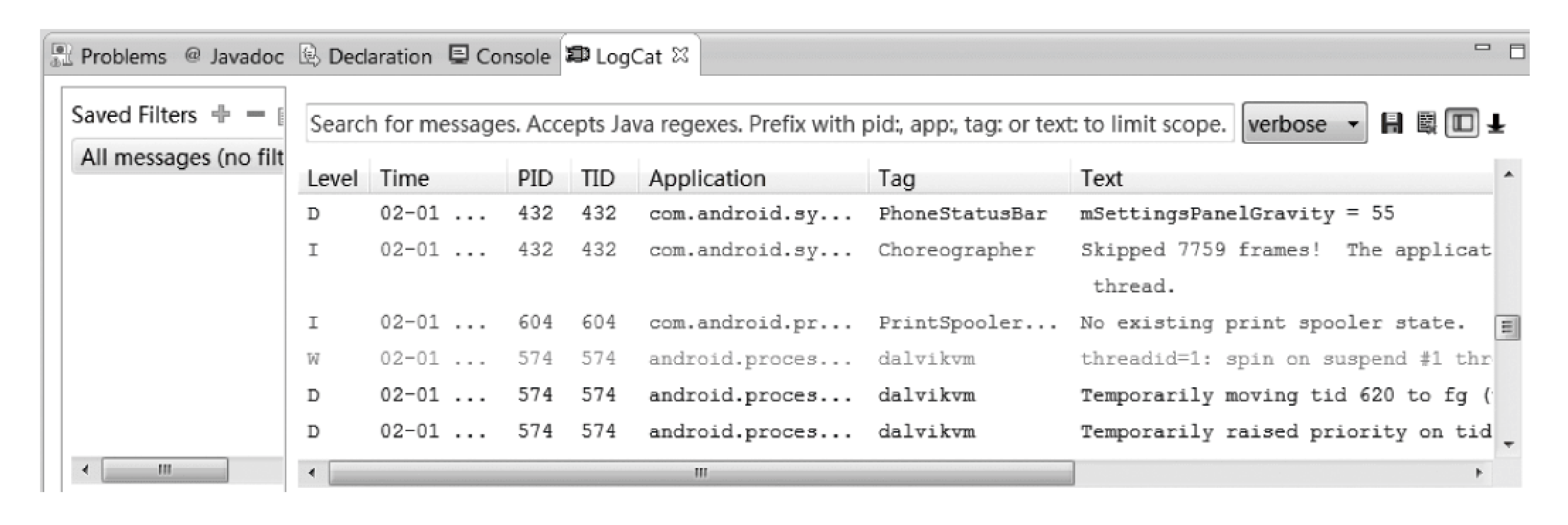


图 2.25 LogCat 视图

1. LogCat 视图

LogCat 视图右上方有一个日志控制级别的下拉列表，前 5 个选项分别为 verbose、debug、info、warn、error，表示 5 种控制级别的信息：详细信息(verbose)、调试信息(debug)、通知信息(info)、警告信息(warn)、错误信息(error)，选项控制级别依次增高。单击某一选项，可以使 LogCat 视图仅输出指定选项的日志信息和高于指定选项控制级别的日志信息，低于指定选项控制级别的日志信息则不显示。

2. 设置 LogCat 过滤器

LogCat 视图提供了过滤功能，单击视图左侧的“＋”按钮，弹出设置 LogCat 过滤器对话

框，如图 2.26 所示，可以增加一个新的过滤器，根据日志信息标签(by Log Tag)，产生日志进程编号(by PID)，日志控制级别(by Log Level)等对显示信息的内容进行过滤。

例如，在图 2.26 中，在 Filter Name 框输入“FilterTest”，在 by Log Tag 框输入“ObservationPoint”，单击 OK 按钮，在 LogCat 视图左边 Saved Filters 框中添加一个新的过滤器 FilterTest，可根据设置的标签 ObservationPoint 对 LogCat 日志输出进行过滤。

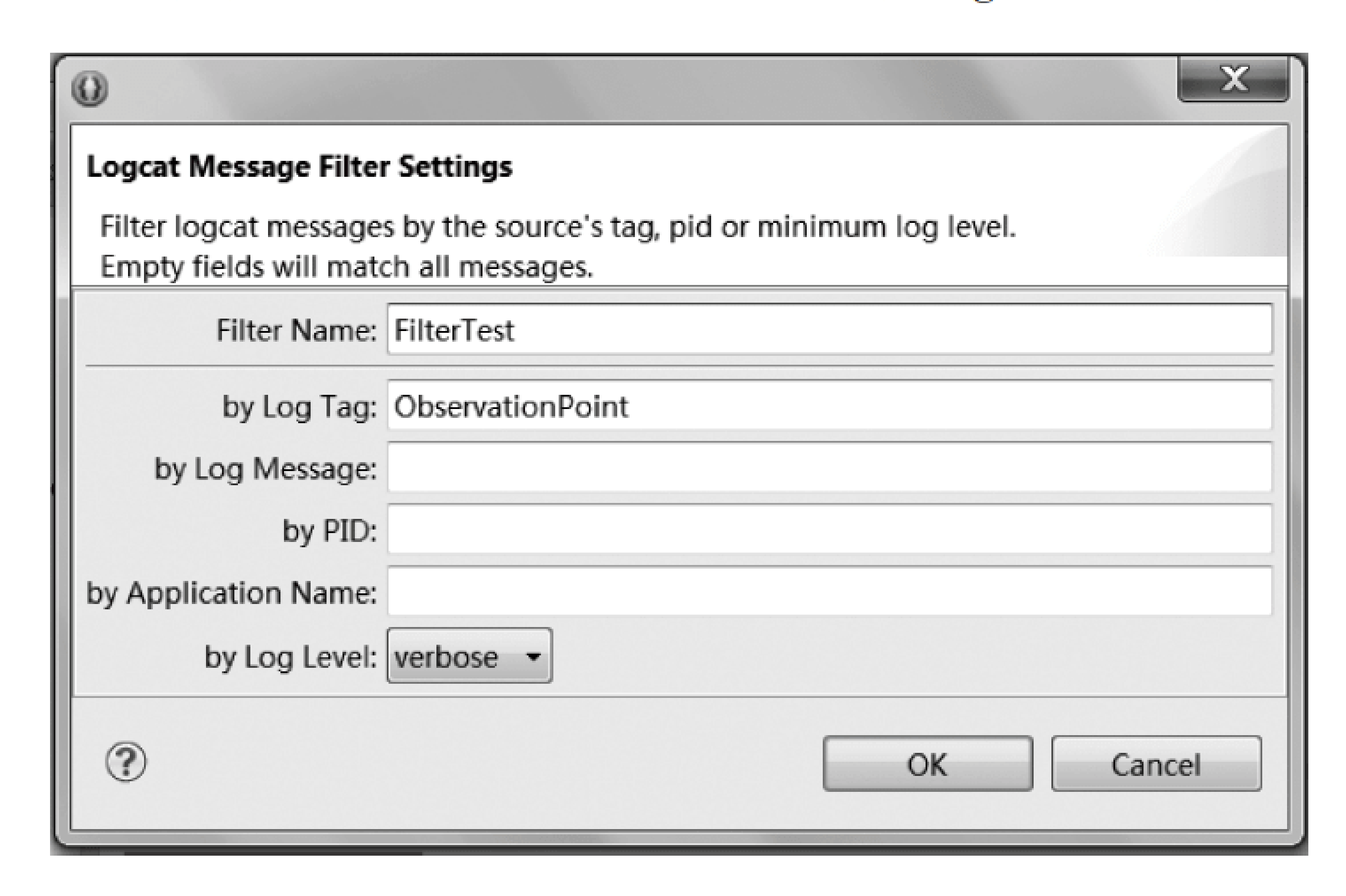


图 2.26　设置 LogCat 过滤器对话框

3. 过滤应用项目的日志信息

LogCat 提供了以下几个过滤输出日志的方法：

(1) Log.v()：用来输出详细信息，对应级别为 verbose，控制级别最低。

(2) Log.d()：用来输出调试信息，对应级别为 debug，比 verbose 高一级。

(3) Log.i()：用来输出通知信息，对应级别为 info，比 debug 高一级。

(4) Log.w()：用来输出警告信息，对应级别为 warn，比 info 高一级。

(5) Log.e()：用来输出错误信息，对应级别为 error，比 warn 高一级。

上述每个方法中都传入两个参数，第一个参数是日志的标签(Tag)，第二个参数是日志要显示的内容。

通过以上方法，可在程序中预先设置一些日志信息，当程序运行时，这些日志信息就会输出到 LogCat 窗口，从而查看应用项目的运行状态。

【例 2.2】 通过应用项目 LogCatDemo 演示过滤日志信息。

应用项目 LogCatDemo 的 MainActivity 类代码如下：

```
package com.application.logcatdemo;

import com.application.logcatdemo.R;
import android.app.Activity;
import android.os.Bundle;
import android.util.Log;

```

```
8  public class MainActivity extends Activity {
9      final static String TAG = "ObservationPoint";
10
11     @Override
12     public void onCreate(Bundle savedInstanceState) {
13         super.onCreate(savedInstanceState);
14         setContentView(R.layout.activity_main);
15
16         Log.v(TAG,"ObservationPoint: Verbose");        //产生一个详细信息
17         Log.d(TAG,"ObservationPoint: Debug");          //产生一个调试信息
18         Log.i(TAG,"ObservationPoint: Info");           //产生一个通知信息
19         Log.w(TAG,"ObservationPoint: Warn");           //产生一个警告信息
20         Log.e(TAG,"ObservationPoint: Error");          //产生一个错误信息
21     }
22 }
```

① 第 6 行使用 import 语句引入“android.util.Log”包文件。

② 第 9 行定义一个用于日志标签的符号常量 ObservationPoint。

③ 第 16 行至第 20 行使用 Log.v()、Log.d()、Log.i()、Log.w()、Log.e()方法在程序中过滤输出日志。当程序运行到以上方法时，预先设置的日志信息便被发送到 LogCat 窗口中。

运行应用项目 LogCatDemo，输出的日志信息如图 2.27 所示。

Declaration　Console　LogCat

Search for messages. Accepts Java regexes. Prefix with pid:, app:, tag: or text: to limit scope.　verbose

Level	Time	PID	TID	Application	Tag	Text
I	02-01...	381	427	system_process	ActivityManager	Start proc com.applicatio o/.MainActivity: pid=871
V	02-01...	871	871	com.application.logcatdemo	ObservationPoint	ObservationPoint: Verbose
D	02-01...	871	871	com.application.logcatdemo	ObservationPoint	ObservationPoint: Debug
I	02-01...	871	871	com.application.logcatdemo	ObservationPoint	ObservationPoint: Info
W	02-01...	871	871	com.application.logcatdemo	ObservationPoint	ObservationPoint: Warn
E	02-01...	871	871	com.application.logcatdemo	ObservationPoint	ObservationPoint: Error
D	02-01...	871	871	com.application.logcatdemo	gralloc_goldfish	Emulator without GPU emul

图 2.27　运行 LogCatDemo 项目的 LogCat 日志信息

可以看出，输出的日志信息太多，迫切需要通过 LogCat 过滤器和控制级别过滤出感兴趣的信息，为此我们已在该项目中编写了 MainActivity 类的代码，并在设置 LogCat 过滤器的例中，设置了一个新的过滤器 FilterTest，其标签(Tag)为 ObservationPoint。

运行应用项目 LogCatDemo，单击 LogCat 视图左边 Saved Filters 框中的过滤器 FilterTest，过滤后的 LogCat 日志信息如图 2.28 所示。

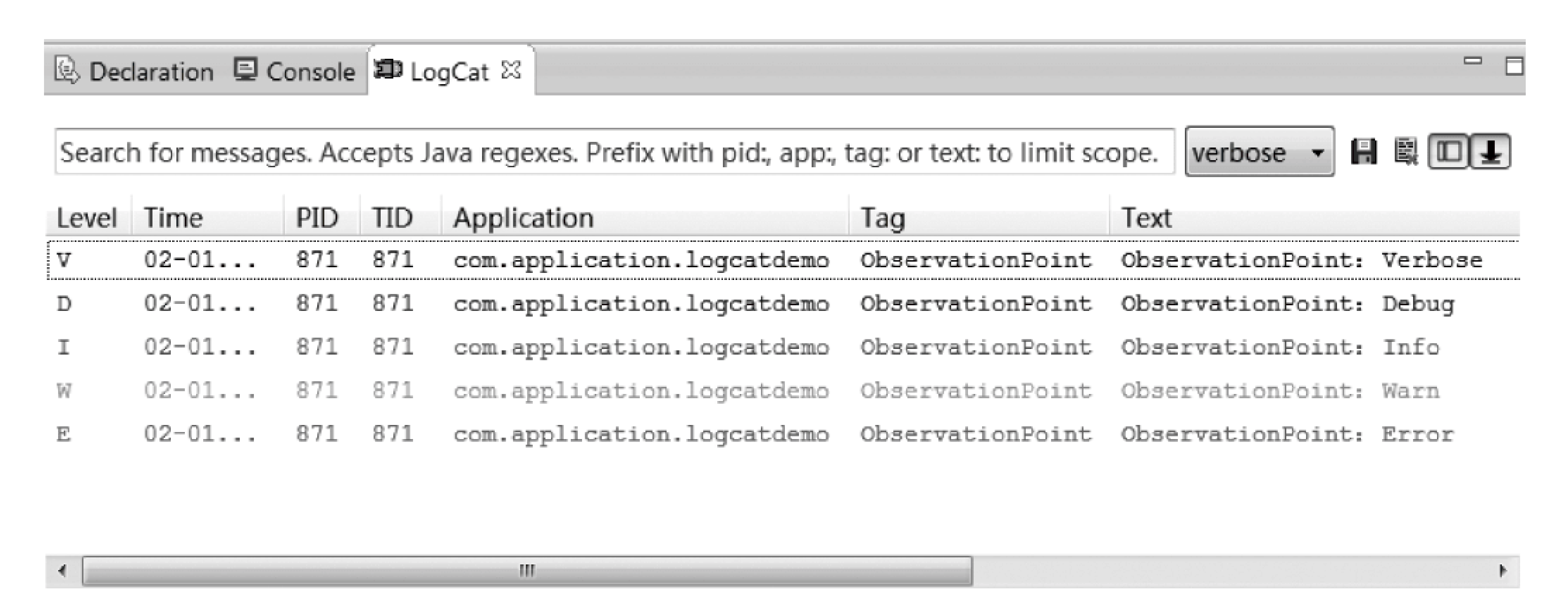


图 2.28　运行 LogCatDemo 项目过滤后的 LogCat 日志信息

2.5　Android 应用项目的发布

运行 Android 应用程序，需要将编译生成后的二进制代码、资源文件和配置文件等打包成 APK(Android Package)文件，即 Android 的安装包，然后将这个 APK 文件发送到模拟器或移动设备上去安装运行，其过程为编译→打包→安装→运行。

发布 Android 应用项目，打包必须使用签名文件对应用项目进行签名，以确定发布者的身份和确保应用的完整性。

在应用项目开发的调试阶段，Eclipse 的 ADT 插件会自动生成调试签名文件对 Android 应用签名，位置在 C:\Users\dell\.android\debug.keystore。当进入应用项目正式发布时，必须使用正式签名文件给应用项目进行签名，不能使用 ADT 插件生成的调试签名文件来发布。

2.5.1　发布 Android 应用项目的打包和签名

发布 Android 应用项目，进行打包和签名的步骤如下：

(1) 在 Eclipse 中，选择需要发布的应用项目，这里选择应用项目 FirstAndroidApplication，右击，在弹出菜单中选择 Android Tools→Export Signed Application Package…，出现 Export Android Application 窗口 Project Checks 页，进入项目校验，如图 2.29 所示。

(2) 在项目校验中，输入选择的项目，这里在 Project 框内输入“FirstAndroidApplication”，单击 Next 按钮，出现 Keystore selection 页，进入签名文件选择，如图 2.30 所示。

(3) 在签名文件选择中，如果没有签名文件，则选中 Create new keystore 单选项，输入签名文件存储路径和密码等信息，这里输入签名文件存储路径 E:\AndroidSigned\demokeystore 和密码，单击 Next 按钮，出现 Key creation 页，进入设置签名文件详细信息，如图 2.31 所示。

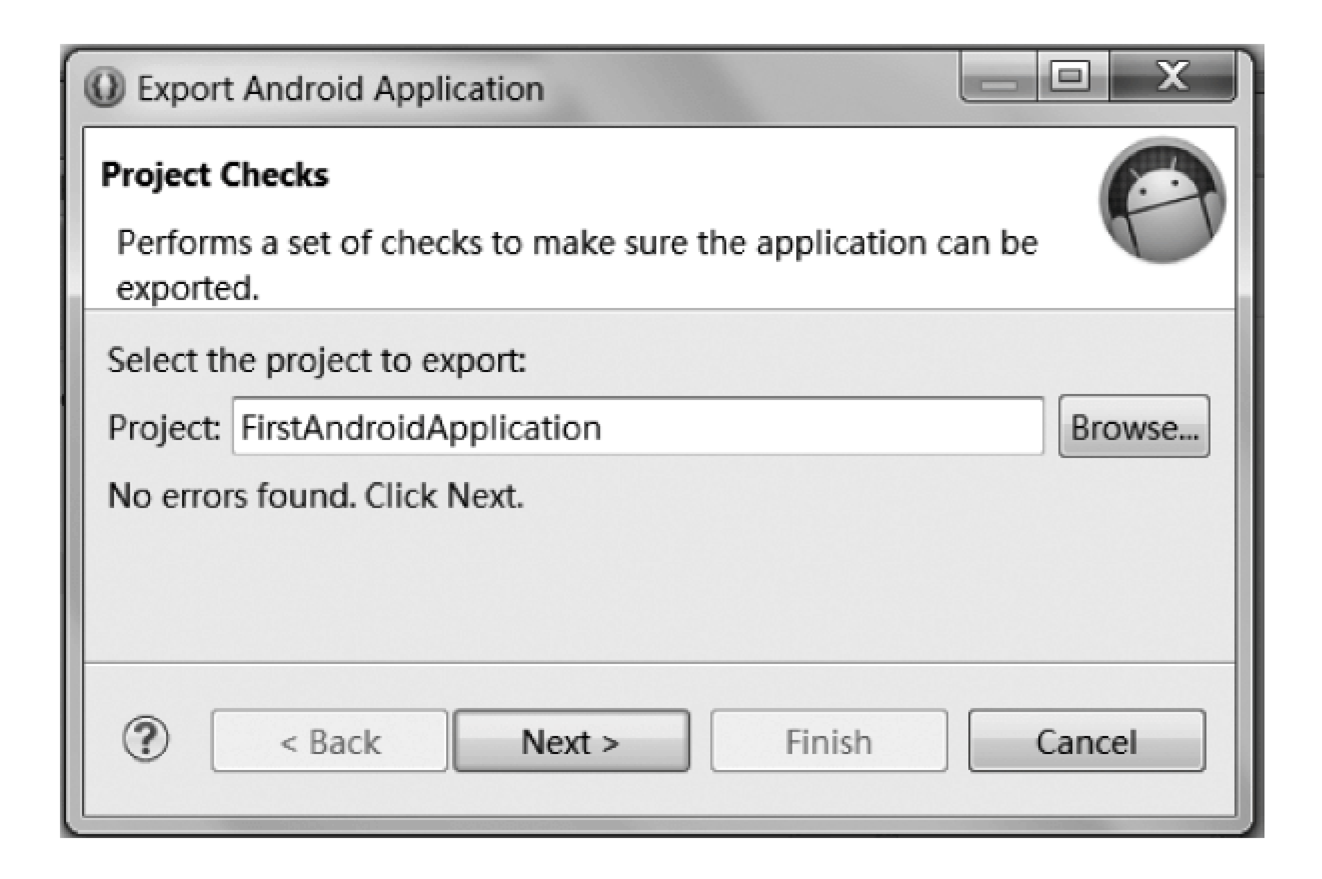


图 2.29　项目校验



图 2.30　签名文件选择

(4) 在设置签名文件详细信息中，输入签名文件别名、密码、有效时间(以年为单位)，以及签名者的相关信息等，这里输入签名文件别名 Test、密码、有效时间 100 年等，单击 Next 按钮，出现 Destination and key/certificate checks 页，进入设置 APK 文件保存路径，如图 2.32 所示。

(5) 在设置 APK 文件保存路径中，输入 APK 文件存储路径，单击 Finish 按钮，生成正式签名后的 APK 文件，打包和签名完毕。

使用 WinRAR 解压软件将打包签名的 APK 文件解压缩，如图 2.33 所示。

可以看到，在 APK 文件中包括配置文件、资源文件、可执行文件和资源目录。

- AndroidMainifest.xml：Android 项目配置文件。
- resources.arsc：Android 资源文件。
- classes.dex：Android Dalvik 可执行文件。



图 2.31　设置签名文件详细信息

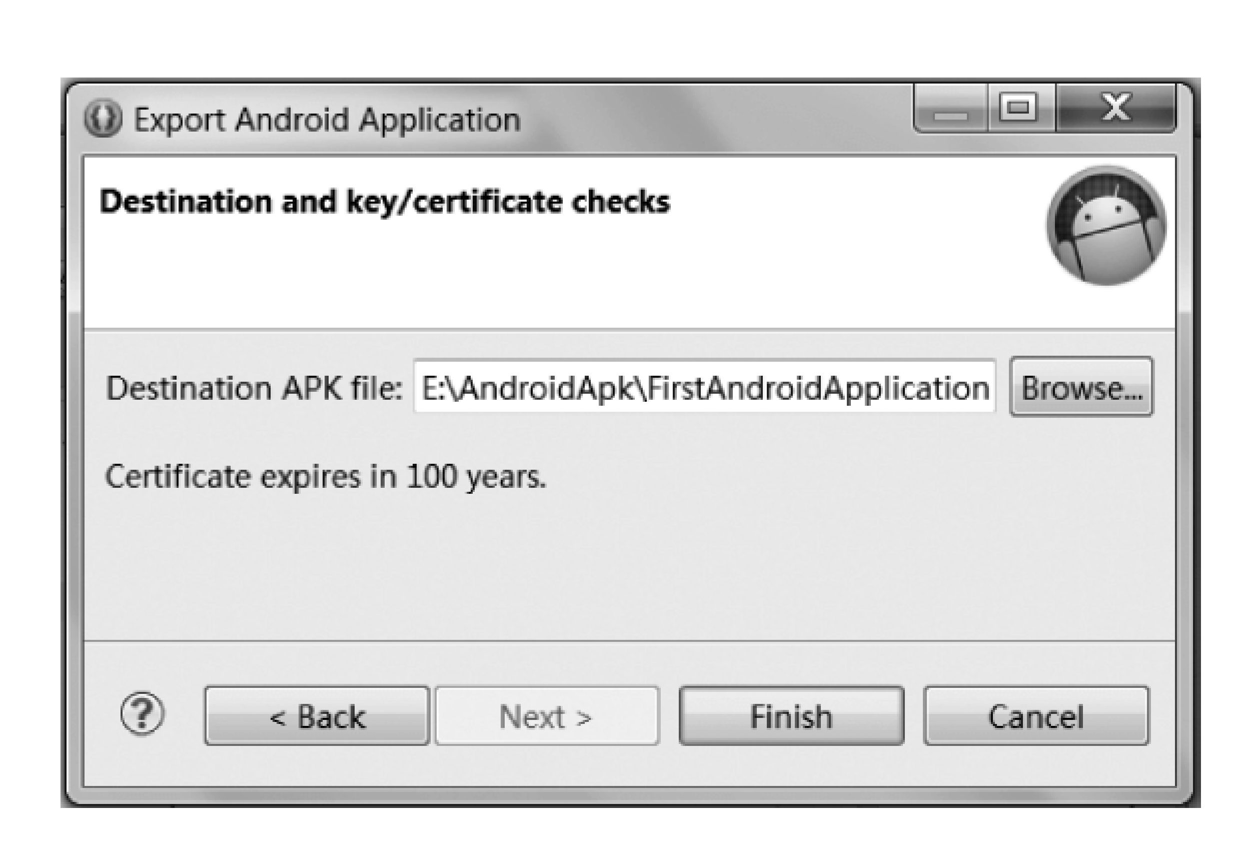


图 2.32　设置 APK 文件保存路径



图 2.33　APK 文件的内容

2.5.2　APK 文件的安装

将 APK 文件安装到移动设备上，有以下两种安装方法。

(1) 将 APK 文件复制到移动设备的 SD 卡中。

使用数据线将移动设备与计算机连接，然后将 APK 文件复制到移动设备的 SD 卡中，

在移动设备中安装运行。

(2) 当移动设备和计算机处于连接状态时，在 Eclipse 中运行应用程序，直接在移动设备中运行。

以 Android 移动设备中的手机为例，其步骤为：下载和安装手机 USB 驱动程序→运行手机调试模式→在手机上运行应用程序，参见 2.1.2 节的内容。

2.6 小　　结

本章主要介绍了以下内容：

(1) 应用 Android 开发平台的结构框架，创建一个新的 Android 应用项目是十分简捷、方便的，介绍了创建第一个 Android 应用项目 FirstAndroidApplication 的步骤。

运行 Android 应用程序，可采用模拟器或移动设备两种方式。

从事 Android 项目开发，时常需要将已完成的项目从 Eclipse 工作区备份到其他机器上，也常常需要借鉴别人已开发好的项目，因此，项目的导出、导入和移除在开发工作中是重要的基本操作。

(2) Android 应用的目录结构包含 src 目录、gen 目录、Android 4.4.2 目录、assets 目录、bin 目录、res 目录、AndroidManifest.xml 文件、proguard-project.txt 文件和 project.properties 文件。

(3) Android 应用开发建立在应用程序框架之上，用户界面由 XML 担任，业务逻辑由 Java 程序完成，以实现界面设计和程序逻辑的分离，具有低耦合性和高重用性，可以让程序设计人员集中精力开发业务逻辑，界面设计人员集中精力进行界面设计，这是符合 MVC 设计思想的，从而达到提高开发效率、缩减开发时间，降低开发成本的目的。

对 src 目录中存放的 MainActivity.java 文件、res 目录下 layout 目录中的 activity_main.xml 文件及 fragment_main.xml 文件和 values 目录中的 strings.xml 文件、gen 目录下的 R.java 文件和项目根目录下 AndroidManifest.xml 文件的代码进行了分析。

(4) 调试是 Android 应用开发的重要工作，介绍了 Android 调试工具：Java 调试器 Debug、图形化调试工具 DDMS 和获取日志信息调试工具 LogCat。

(5) 运行 Android 应用程序，需要将编译生成后的二进制代码、资源文件和配置文件等打包成 APK(Android Package)文件，即 Android 的安装包，然后将这个 APK 文件发送到模拟器或移动设备上去安装运行，其过程为编译→打包→安装→运行。发布 Android 应用项目，打包必须使用签名文件对应用项目进行签名，以确定发布者的身份和确保应用的完整性。

介绍了发布 Android 应用项目，进行打包和签名的步骤。

将 APK 文件安装到移动设备上，有两种安装方法：一种是将 APK 文件复制到移动设备的 SD 卡中，在移动设备中安装运行；一种是当移动设备和计算机处于连接状态时，在 Eclipse 中运行应用程序，直接在移动设备中运行。

习　题　2

一、选择题

2.1　Android 应用的目录结构未包含的目录是________。

A. src　　B. assets　　C. webroot　　D. res

2.2　Android 应用的根目录下未包含的文件是________。

A. AndroidManifest. xml　　B. web. xml

C. project. properties　　D. proguard-project. txt

2.3　APK 文件在哪一个目录下？ ________

A. src　　B. gen　　C. res　　D. bin

2.4　assets 目录的作用是________。

A. 放置多媒体数据文件　　B. 放置应用到的图片资源

C. 放置与 UI 相应的布局文件　　D. 放置字符串、颜色、数组等常量数据

2.5　关于包(Package)的说法错误的是________。

A. 包名可作为 Android 应用的唯一标识

B. 文件的组织方式

C. 一个管理名字空间的机制

D. 每个包对应一个目录结构

2.6　下面关于继承(Inheritance)的叙述不正确的是________。

A. 子类可以继承父类的成员变量和成员方法

B. 可以实现代码的复用

C. 子类可以修改父类的成员变量或重写父类的方法

D. 一个子类可以有多个父类

2.7　在重写(Override)中________。

A. 子类方法名称可以和父类不同

B. 子类方法参数个数可以和父类不同

C. 子类重新定义自己的方法

D. 子类方法参数类型可以和父类不同

2.8　关于 R. java 文件叙述不正确的是________。

A. 由 Android 自动生成

B. 为项目中的各个资源创建其唯一的 ID

C. 可以通过 ID 得到该资源的引用

D. 用户可以修改

2.9　关于 AndroidManifest. xml 文件叙述不正确的是________。

A. 存放在项目 res 目录下

B. 在程序中定义组件需要在这里注册

C. 每个 Android 项目必需的文件

D. 项目的系统控制文件

2.10 在 APK 中的文件不包括________。

A. resources. arsc B. classes. dex

C. MainActivity. java D. AndroidMainifest. xml

二、填空题

2.11 res 目录存放整个项目所用的全部资源文件，包括所有图形、布局和________资源等。

2.12 在 Android 应用开发中，用户界面由________担任，业务逻辑由 Java 程序完成。

2.13 在 src 目录中存放应用程序________源代码文件，自动地组织在用户声明的包内。

2.14 layout 目录存放应用程序的________。

2.15 Android 调试工具有 Java 调试器 Debug、图形化调试工具 DDMS 和获取日志信息调试工具________。

2.16 在 LogCat 提供的过滤输出日志的方法中，Log. e()用来输出________。

2.17 运行 Android 应用程序的过程为________。

三、问答题

2.18 Android 应用的目录结构包含哪些目录？各个目录有何作用？

2.19 Android 有哪些调试工具？各有何功能？

四、应用题

2.20 新建一个 Android 应用项目，取名为 FirstProject，将运行结果“Hello world!”替换为“第一个 Android 应用项目!”。

2.21 进行 Android 项目的导入、导出和移除等上机实验。

2.22 使用 LogCat 查看应用项目的日志信息，新建过滤器过滤日志中的错误信息。

2.23 对应用项目 FirstProject，进行打包、签名、安装、运行等上机实验。

第3章 Activity、Fragment 和 Intent

本章要点

- Android 应用程序的生命周期为从启动到终止的全过程,由系统进行调度和控制。
- Android 应用的基本组件有 Activity(活动)、Service(服务)、BroadcastReceiver(广播接收器)、ContentProvider(数据提供器)、Intent(意图)等。
- Activity 的生命周期中存在启动状态、运行状态、暂停状态、停止状态、销毁状态 5 种状态。
- Activity 的生命周期可分为完全生命周期、可视生命周期和活动生命周期,每种生命周期中包含不同的回调方法。
- Fragment 不能独立存在,它必须嵌入到 Activity 中,Fragment 的生命周期被其所属的 Activity 生命周期控制。
- Intent 用于启动 Activity、Service 或者 BroadcastReceiver 等组件,并且是组件之间通信的重要媒介。
- Intent 对象包含 Component、Action、Data、Category、Extra 及 Flag 等 6 种属性。

Android 应用程序由 Activity、Service、BroadcastReceiver、ContentProvider 等组件构成,Activity 组件为用户提供可视化用户界面,它是 Android 应用程序中最常见、最基本的组件。本章介绍 Android 应用程序的生命周期,Android 应用的基本组件,Activity 的运行状态和生命周期,Fragment 的使用,Intent 的组成、调用和传递数据等内容。

3.1 Android 应用程序的生命周期

Android 应用程序生命周期指从启动到终止的全过程,应用程序的生命周期是由 Android 系统进行调度和控制,而不是由应用程序直接控制的。

Android 应用程序组件有其生命周期,指从创建到销毁的全过程,组件会在可见、不可见、活动、不活动等状态中不断变化,Activity 组件是 Android 应用生命周期的重要部分之一。

1. 进程

进程(Process)是程序的一次执行,进程由程序、数据和进程控制块构成,进程是一个可拥有资源的独立实体,又是一个可以独立调度的基本单位。

进程的执行过程包括创建(New)、就绪(Ready)、执行(Running)、阻塞(Blocked)、挂起(Suspend)、终止(Terminated)等状态。

在 Android 操作系统中,进程是应用程序的具体实现。组件运行的进程由

Androidmanifest 文件控制。组件标签<activity>、<service>、<receiver>、<provider>等包含一个 process 属性，这个属性可以设置组件运行的进程。<application>标签也包含 process 属性，用来设置程序中所有组件的默认进程。所有的组件在默认进程的主线程中实例化，系统对这些组件的调用从主线程中分离。

每个 Android 应用程序的进程都是由 Android 进行时独立管理的，每个 Android 的应用程序在自己的进程中运行。

Android 系统往往运行在资源受限的平台上，资源管理非常重要，因此，由 Android 系统管理资源。

Android 系统的进程优先级从高到低分别为：前台进程、可见进程、服务进程、后台进程、空进程。前台进程为高优先级、可见进程、服务进程为中优先级、后台进程、空进程为低优先级。

1）前台进程

前台进程是 Android 系统中最重要的进程，它是与用户进行交互的进程。

前台进程包括：

- 该进程拥有一个正在与用户交互的 Activity()（其 onResume()方法被调用）。
- 该进程拥有一个绑定到正与用户交互的 Activity 上的 Service。
- 该进程拥有一个前台运行并调用了 startForeground()方法的 Service。
- 该进程拥有一个正在执行的回调方法（如 onStart()、onCreate()、onDestroy()）的 Service。
- 该进程拥有一个正在执行 onReceive()方法的 BroadcastReceiver 对象。

通常在任何时间点，只有很少前台进程存在。当出现资源不足时，也会“杀死”部分前台进程。

2）可见进程

可见进程是用户能够在屏幕上看见，但不能与用户进行交互，不响应界面事件的进程。

可见进程包括：

- 该进程拥有一个不在前台但为用户可见的 Activity（如调用了方法 onPause()之后）。
- 一个可见的 Activity 所绑定的 Service。

当出现无法维持前台进程运行等情况时，才会清除可见进程。

3）服务进程

包含已启动服务的进程称为服务进程。服务进程不可见，不与用户直接交互，但能在后台运行，提供用户需要的功能。

服务进程包括：

- 一个由 startService()方法启动的 Service。
- 支持正在处理的不需要可见界面运行的 Service。

当系统内存不足，不能维持前台进程和可见进程的运行时，才会清除服务进程。

4）后台进程

不包含任何已启动服务，而且没有用户可见的 Activity 的进程，即为后台进程。

后台进程包括：

- 该进程拥有一个当前不可见的 Activity(已调用了 onStop()方法)。
- 目前没有服务的 Service。

一般情况下，存在较多的后台进程，当系统资源紧张时，Android 将会使用 LRU 模式来清除最近最少使用的后台进程。

5) 空进程

空进程是不包含任何 Activity 组件，对用户没有任何作用的进程。

为了改善系统的整体性能，Android 通常在内存中保留生命周期结束了的应用，当系统资源紧张时，空进程首先被清除。

2. 线程

线程(Thread)是进程中的一个实体，是被系统独立调度的基本单位。线程基本上不拥有系统资源，只有一些在运行中必不可少的资源(如程序计数器、一组寄存器和栈)，但它可共享所属进程的全部资源。

引入线程的目的是为了减少程序并发执行时所付出的时空开销，使操作系统具有更好的并发性，提高系统运行的效率。线程具有许多传统进程所具有的特征，又称为轻量级进程(Light-Weight Process)，而把传统的进程称为重量级进程(Heavy-Weight Process)。

每个进程有一到多个线程运行在其中。进程中的所有组件都在 UI 线程中实例化，以保证应用程序是单线程的，除非应用程序又创建了自己的线程，例如网络连接、下载或其他费时操作。线程通过 Java 的 Thread 类创建。

3.2 Android 应用的基本组件

Android 应用程序由组件组成，并通过项目的 AndroidManifest.xml 将它们绑定在一起。

Android 应用中常用的基本组件有 Activity(活动)、Service(服务)、BroadcastReceiver(广播接收器)、ContentProvider(数据提供器)、Intent(意图)等，下面分别进行介绍。

3.2.1 Activity

Activity 用于提供可视化用户界面并与用户交互，它是最常用的组件，Activity 是应用程序的显示层，显示可视化的用户界面，并接收与用户交互所产生的界面事件。

一个 Activity 展现一个可视化用户界面，如果需要多个可视化用户界面，该 Android 应用会包含多个 Activity，尽管多个 Activity 在一起工作，但每个 Activity 是相对独立的，每个 Activity 都继承自 android.app.Activity 类。

例如，第一个 Android 应用项目 FirstAndroidApplication 中 MainActivity.java 的代码如下：

```
...
import android.app.Activity;
import android.view.View;
import android.view.ViewGroup;
```

```
…
public class MainActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
…
    }
…
```

Activity 的显示内容由 View(视图)组件的对象提供,并定义在 res/layout 下的 XML 文件中,View 组件的对象包括文本框、多选框、单选框、按钮、菜单等。

通过 Activity 将指定的 View 显示出来,调用 Activity 的 setContentView()方法,例如上面代码中的 setContentView(R.layout.activity_main)方法。

3.2.2 Service

Service 是一个常用组件,需要继承 Service 类。Service 一般用于没有用户界面,又需要长时间在后台运行的应用,例如播放背景音乐或在网络上获取数据。

Service 与 Activity 有以下区别:Service 通常位于后台运行,它一般不需要与用户交互,也没有用户界面。Service 一般由 Activity 启动,但拥有自己独立的生命周期。Service 具有较长的生命周期,当启动它的 Activity 生命周期结束,Service 仍能继续运行,直到自己的生命周期结束。

Service 有两种启动方式:

(1) 使用 startService 方式启动。

(2) 使用 bindService 方式启动。

3.2.3 BroadcastReceiver

BroadcastReceiver 是另一个常用组件,用来接收广播消息,不包含任何用户界面,其监听的事件源是其他组件。

使用 BroadcastReceiver 组件接收广播信息,需要继承 BroadcastReceiver 类并重写 onReceive 方法。当其他组件通过 sendBroadcast()、sendStickyBroadcast()或 sendOrderBroadcast()方法发送广播消息时,如果通过 IntentFilter 过滤的 BroadcastReceiver 感兴趣,就会被接收。

BroadcastReceiver 注册方式有两种:

(1) 在 AndroidManifest.xml 中,在<receiver></receiver>标签中设置。

(2) 在 Java 代码中,通过 Context.registReceiver()方法注册。

3.2.4 ContentProvider

ContentProvider 组件是 Android 系统提供的一种标准的共享数据的机制,用来管理和共享应用程序的数据存储。例如开发一个发送短信的程序,需要多个应用程序之间共享和

交换数据。

一般的使用方法是：一个应用程序使用 ContentProvider 暴露自己的数据，另一个应用程序使用 ContentResolver 访问数据。

3.2.5 Intent

Intent 是不同组件间通信的载体，是连接各个组件的桥梁。Intent 不仅可以用到不同组件之间的交互，还可以用到不同应用程序之间的交互。

Activity、Service、BroadcastReceiver 组件之间的通信都使用 Intent 作为通信的载体，但各个组件使用 Intent 的机制不同。

(1) 当需要启动一个 Activity 时，可调用 Context.startActivity()或 Context.startActivityForResult()方法，这两个方法中的 Intent 参数封装了需要启动的目标 Activity 的信息。

(2) 当需要启动一个 Service 时，可调用 Context.startService()或 Context.bindService()方法，这两个方法中的 Intent 参数封装了需要启动的目标 Service 的信息。

(3) 当需要触发一个 BroadcastReceiver 时，可调用 sendBroadcast()、sendStickyBroadcast()或 sendOrderedBroadcast()方法，这三个方法中的 Intent 参数封装了需要触发的目标 BroadcastReceiver 的信息。

3.3 Activity 的运行状态和生命周期

Activity 生命周期指 Activity 从启动到销毁的过程，下面介绍 Activity 的运行状态和生命周期。

3.3.1 Activity 的运行状态

Activity 的生命周期中存在五种状态：启动状态、运行状态、暂停状态、停止状态、销毁状态。

(1) 启动状态(Starting)：Activity 在屏幕的前台。

(2) 运行状态(Running)：Activity 可见，获得焦点，可与用户进行交互。Activity 启动后，随即进入运行状态。

(3) 暂停状态(Paused)：Activity 失去焦点，但仍可见，依然保持活力，但在系统内存极低时将被杀掉。

(4) 停止状态(Stopped)：Activity 失去焦点，不可见，此时 Activity 被另一个 Activity 完全覆盖，系统可以随时将其释放。

(5) 销毁状态(Destroyed)：系统将 Activity 从内存中删除，有两种方式，一种是要求该 Activity 结束，一种是直接被杀掉。

3.3.2 Activity 的生命周期

本节介绍 Activity 生命周期和回调方法。

1. Android 的回调机制

一个通用的程序架构具有完成整个应用的流程和功能，但在某个特定点需要一段业务相关的代码进行处理，例如 Activity 的 onCreate()、onPause()和 onStop()等回调方法，开发人员可以选择性地重写这些方法，通用的程序架构就会回调该方法进行相关的业务处理。

2. Activity 的回调方法

Activity 的回调方法有 onCreate()、onStart()、onResume()、onPause()、onStop()、onRestart()、onDestroy()、onSaveInstanceState()、onRestoreInstanceState()等，下面分别介绍。

(1) onCreate(Bundle)：

创建 Activity 时被回调，该方法只会被调用一次。如果 Activity 之前是被冻结状态，其状态由 Bundle 提供，接收参数为 null 或由 onSaveInstanceState()方法保存的状态信息。其后调用 onStart()或 onRestart()方法。

(2) onStart()：

启动 Activity 时被回调，当 Activity 对用户即将可见时被调用。

(3) onResume()：

恢复 Activity 时被回调，当 Activity 可以开始与用户进行交互之前被调用。

(4) onPause()：

暂停 Activity 时被回调，活动将进入后台时会运行该方法，当系统将要启动另一个 Activity 之前被调用。

(5) onStop()：

停止 Activity 时被回调，当 Activity 不再为用户可见时被调用。

(6) onRestart()：

重新启动 Activity 时被回调，在再次启动之前被调用。

(7) onDestroy()：

销毁 Activity 时被回调，在 Activity 销毁前被调用。

(8) onSaveInstanceState(Bundle)：

回调该方法让活动可以保存每个实例的状态。

(9) onRestoreInstanceState(Bundle)：

回调 onSaveInstanceState()方法保存的状态来重新初始化某个活动时调用该方法，其后紧跟的方法是 onResume()。

3. Activity 的生命周期

Activity 的生命周期如图 3.1 所示。

Activity 的生命周期可分为完全生命周期、可视生命周期和活动生命周期，每种生命周期中包含不同的回调方法，如图 3.2 所示。

1) 完全生命周期

完全生命周期是从 Activity 创建到销毁的全部过程，从调用 onCreate()开始到 onDestroy()结束。开发人员通常在 onCreate()中初始化 Activity 所能使用的全局资源和状态，并在 onDestroy()中释放这些资源。在一些极端的情况下，Android 系统不调用 onDestroy()，直接终止进程。

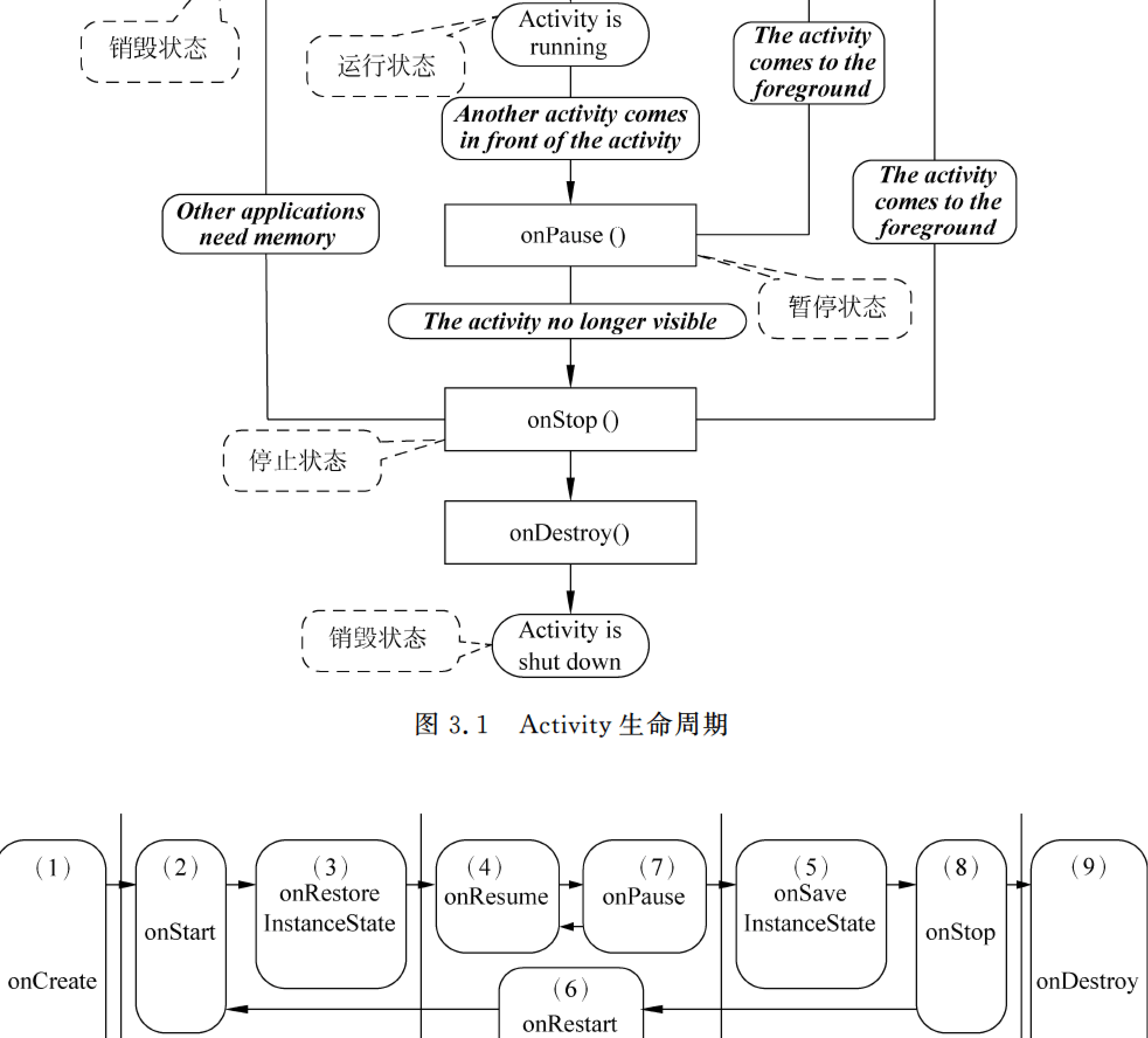


图 3.1 Activity 生命周期

图 3.2 Activity 生命周期分类

2）可视生命周期

可视生命周期是 Activity 在界面上从可见到不可见的过程，从调用 onStart()开始到 onStop()结束。onStart()一般用来初始化或启动与更新界面相关的资源，onStop()一般用来暂停或停止一切与更新用户界面相关的线程、计时器和服务。onRestart()在 onSart()前被调用，用来在 Activity 从不可见变为可见的过程中，进行一些特定的处理过程。onStart()和 onStop()会被多次调用，使 Activity 不断地从可见到不可见，再从不可见到可见。

3）活动生命周期

活动生命周期是 Activity 在屏幕的最上层，并能够与用户交互的阶段，从调用 onResume()开始到 onPause()结束。在 Activity 的状态变换过程中 onResume()和 onPause()经常被调用，因此这两个回调方法中应使用简单、高效的轻量级代码。

【例 3.1】 为了更好地理解 Activity 生命周期和 Android 的回调机制，通过 LifeCycle 示例的演示进行说明和分析。

【解题思路】

通过在生命周期回调方法中添加“日志点”的方法进行调试，程序的运行结果将会显示在 LogCat 中。为了使显示结果易于观察和分析，在 LogCat 中设置过滤器 LifeTest，过滤方法选择 by Log Tag，过滤关键字为 ActivityLifeCycle。

【开发步骤和程序分析】

(1) 在 Eclipse 中创建一个 LifeCycle 应用项目，包名为 com. application. lifecycle。

(2) 在 src/com. application. lifecycle 包下的 lifecycle. java 文件中，加载 main. xml 布局文件，在生命周期回调方法中添加“日志点”。

在该文件中编辑代码如下：

```
package com.application.lifecycle;

import android.app.Activity;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.Button;


    public class LifeCycle extends Activity {
    private static String TAG = "ActivityLifeCycle";

    @Override                          //生命周期开始，创建 Activity
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        Log.i(TAG, " -- (1) onCreate()");

        Button button = (Button)findViewById(R.id.btn_finish);
        button.setOnClickListener(new View.OnClickListener() {
```

```
            public void onClick(View view) {
                finish();
            }
        });
    }

    @Override                          //启动 Activity
    public void onStart() {
        super.onStart();
        Log.i(TAG, " -- (2) onStart()");
    }

    @Override                          //重新初始化 Activity
    public void onRestoreInstanceState(Bundle savedInstanceState) {
        super.onRestoreInstanceState(savedInstanceState);
        Log.i(TAG, " -- (3) onRestoreInstanceState()");
    }

    @Override                          //恢复 Activity
    public void onResume() {
        super.onResume();
        Log.i(TAG, " -- (4) onResume()");
    }

    @Override                          //让 Activity 保存实例的状态
    public void onSaveInstanceState(Bundle savedInstanceState) {
        super.onSaveInstanceState(savedInstanceState);
        Log.i(TAG, " -- (5) onSaveInstanceState()");
    }

    @Override                          //重新启动 Activity
    public void onRestart() {
        super.onRestart();
        Log.i(TAG, " -- (6) onRestart()");
   }

    @Override                          //暂停 Activity
    public void onPause() {
        super.onPause();
        Log.i(TAG, " -- (7) onPause()");
    }

    @Override                          //停止 Activity
    public void onStop() {
        super.onStop();
```

```
        Log.i(TAG, " -- (8) onStop()");
    }

    @Override                          //生命周期结束,销毁 Activity
    public void onDestroy() {
        super.onDestroy();
        Log.i(TAG, " -- (9) onDestroy()");
    }
}
```

【运行结果】

在 Eclipse 中启动模拟器,然后运行项目 LifeCycle,运行结果如图 3.3 所示。



图 3.3　LifeCycle 应用项目界面

(1) 演示完全生命周期。

启动项目 LifeCycle 后,单击项目界面中的"结束"按钮,LogCat 输出结果如图 3.4 所示。

Search for messages. Accepts Java regexes. Prefix with pid:, app:, tag: or text: to limit scope.　verbose

Level	Time	PID	TID	Application	Tag	Text
I	02-20 21:09:00.210	1224	1224	com.application.lifecycle	ActivityLifeCycle	--(1) onCreate()
I	02-20 21:09:00.210	1224	1224	com.application.lifecycle	ActivityLifeCycle	--(2) onStart()
I	02-20 21:09:00.210	1224	1224	com.application.lifecycle	ActivityLifeCycle	--(4) onResume()
I	02-20 21:09:09.310	1224	1224	com.application.lifecycle	ActivityLifeCycle	--(7) onPause()
I	02-20 21:09:11.130	1224	1224	com.application.lifecycle	ActivityLifeCycle	--(8) onStop()
I	02-20 21:09:11.130	1224	1224	com.application.lifecycle	ActivityLifeCycle	--(9) onDestroy()

图 3.4　Activity 完全生命周期回调方法次序

由图 3.4 可看出,回调方法的调用顺序如下:(1)onCreate→(2)onStart→(4)onResume→(7)onPause→(8)onStop→(9)onDestroy。

启动 Activity 时,系统首先调用 onCreate()分配资源,再调用 onStart()将 Activity 显示在屏幕上,然后调用 onResume()获取焦点,能够与用户进行交互,此时用户能够正常使用这个 Android 项目。

当用户单击"结束"按钮时,系统相继调用 onPause()、onStop()和 onDestroy(),释放资源并销毁进程。

(2) 演示可视生命周期。

正常启动 LifeCycle，再通过“Call 键”(拨号键)启动内置的拨号程序，然后通过“Back 键”(回退键)退出拨号程序，LifeCycle 重新显示在屏幕上，LogCat 输出结果如图 3.5 所示。

Search for messages. Accepts Java regexes. Prefix with pid:, app:, tag: or text: to limit scope. verbose

Level	Time	PID	TID	Application	Tag	Text
I	02-20 21:43:...	1365	1365	com.application...	ActivityLifeCycle	--(1) onCreate()
I	02-20 21:43:...	1365	1365	com.application...	ActivityLifeCycle	--(2) onStart()
I	02-20 21:43:...	1365	1365	com.application...	ActivityLifeCycle	--(4) onResume()
I	02-20 21:43:...	1365	1365	com.application...	ActivityLifeCycle	--(7) onPause()
I	02-20 21:43:...	1365	1365	com.application...	ActivityLifeCycle	--(5) onSaveInstanceState()
I	02-20 21:43:...	1365	1365	com.application...	ActivityLifeCycle	--(8) onStop()
I	02-20 21:44:...	1365	1365	com.application...	ActivityLifeCycle	--(6) onRestart()
I	02-20 21:44:...	1365	1365	com.application...	ActivityLifeCycle	--(2) onStart()
I	02-20 21:44:...	1365	1365	com.application...	ActivityLifeCycle	--(4) onResume()

图 3.5　Activity 可视生命周期回调方法次序

由图 3.5 可看出，回调方法的调用顺序：(1)onCreate→(2)onStart→(4)onResume→(7)onPause→(5)onSaveInstanceState→(8)onStop→(6)onRestart→(2)onStart→(4)onResume。

Activity 启动时，回调方法的调用顺序仍为(1)onCreate→(2)onStart→(4)onResume。

当按下“Call 键”(拨号键)时，内置拨号程序被启动，原有的 Activity 被覆盖，系统首先调用 onPause()，再调用 onSaveInstanceState()保存 Activity 状态，最后调用 onStop()停止对不可见的 Activity 的更新。

当按下“Back 键”(回退键)时，退出拨号程序，系统调用 onRestart()恢复界面上需要更新的信息，再调用 onStart()和 onResume()重新显示 Activity，能够与用户进行交互。

3.4　Fragment 的使用

Fragment(片段)以 Activity 界面的一个组成部分出现。

3.4.1　Fragment 的生命周期

Fragment 有自己的生命周期，但它的生命周期受其所在的 Activity 生命周期控制，Fragment 不能独立存在，它必须嵌入到 Activity 中。当 Activity 暂停时，它拥有的所有的 Fragment 都暂停了；当 Activity 销毁时，它拥有的所有 Fragment 都被销毁；当 Activity 处于活动状态时(在 onResume()之后，onPause()之前)，用户可以通过方法操作每个 Fragment。Fragment 的生命周期如图 3.6 所示。

Fragment 有以下特点：

- Fragment 总是作为 Activity 界面的组成部分。
- Fragment 有自己的生命周期，但它的生命周期被其所属的 Activity 生命周期控制。
- 在 Activity 运行过程中，可动态地添加、删除和替换 Fragment。

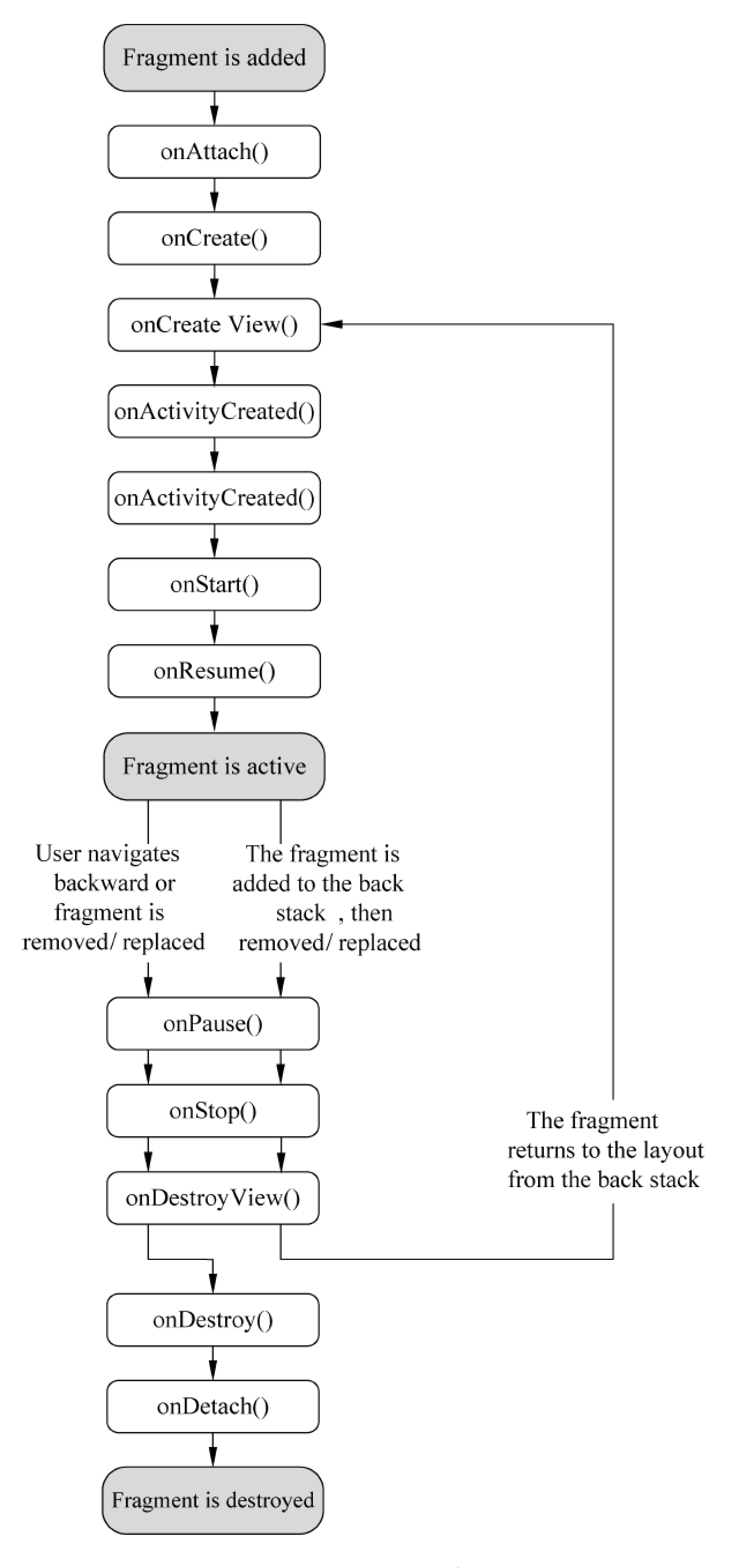


图 3.6　Fragment 生命周期

- 一个 Activity 中可同时出现多个 Fragment，一个 Fragment 也可在多个 Activity 中使用。
- Fragment 可以响应自己的输入事件。

1. Fragment 对象跟用户交互时需要回调的方法

1）onAttach(Activity)

当 Fragment 对象跟 Activity 关联时，调用该方法。

2）onCreate(Bundle)

当 Fragment 对象初始创建时，调用该方法。

3）onCreateView(LayoutInflater, ViewGroup, Bundle)

该方法用于创建和返回跟 Fragment 关联的 View 对象。

4）onActivityCreate(Bundle)

该方法会告诉 Fragment 对象，它所依附的 Activity 对象已经完成了 Activity.onCreate()方法的执行。

5）onStart()

该方法会让 Fragment 对象显示给用户（在包含该 Fragment 对象的 Activity 被启动后）。

6）onResume()

该方法会让 Fragment 对象跟用户交互（在包含该 Fragment 对象的 Activity 被启动恢复后）。

2. Fragment 对象不再使用时需要回调的方法

1）onPause()

当 Fragment 对象所依附的 Activity 对象被挂起，或者在 Activity 中正在执行一个修改 Fragment 对象的操作，而导致 Fragment 对象不再跟用户交互时，系统会调用该方法。

2）onStop()

当 Fragment 对象所依附的 Activity 对象被终止，或者在 Activity 中正在执行一个修改 Fragment 对象的操作，而导致 Fragment 对象不再显示给用户时，系统会调用该方法。

3）onDestroyView()

该方法用于清除跟 Fragment 中的 View 对象关联的资源。

4）onDestroy()

当 Fragment 对象的状态被最终清理完成之后，要调用该方法。

5）onDetach()

当 Fragment 对象不再跟它依附的 Activity 关联的时候，该方法会立即被调用。

3.4.2 Fragment 的应用

Fragment 在应用中是一个模块化和可重用的组件，下面介绍向 Activity 中添加 Fragment 的方法、Fragment 常用的类和方法、Fragment 的子类等内容。

1. 向 Activity 中添加 Fragment 的方法

向 Activity 中添加 Fragment 有两种方法：一种是直接在布局文件中添加，另一种是当 Activity 运行时添加。

1）直接在布局文件中添加 Fragment

直接在布局文件中添加 Fragment，可以使用<fragment>标记实现，将 Fragment 作为

Activity 整个布局的一部分。

2）当 Activity 运行时添加 Fragment

当 Activity 运行时，也可以将 Fragment 添加到 Activity 的布局中，实现方法是获取一个 FragmentTransaction 的实例，然后使用 add()方法添加一个 Fragment，再调用 commit()方法提交事务。

2. Fragment 常用的类

（1）android. app. Fragment：用于定义 Fragment。

（2）android. app. FragmentManager：用于在 Activity 中操作 Fragment。通过调用 Activity 的 getFragmentManager()方法可以取得 FragmentManager 的实例。

（3）android. app. FragmentTransaction：对 Fragment 进行添加、移除、替换及执行其他动作。

在使用 FragmentTransaction 的方法前，首先需要取得 FragmentManager 的实例，再利用 FragmentManage 的 beginTransaction()方法开启一个事务，获取一个 FragmentTransaction 对象。

3. FragmentTransaction 的方法

（1）FragmentTransaction. add()：往 Activity 中添加一个 Fragment。

（2）FragmentTransaction. remove()：从 Activity 中移除一个 Fragment，如果被移除的 Fragment 没有添加到回退栈，这个 Fragment 实例将会被销毁。回退栈（back stack）由 Activity 管理，允许用户通过按下 Back 按键返回到前一个 Fragment 状态。

（3）FragmentTransaction. replace()：使用另一个 Fragment 替换当前的 Fragment。

（4）FragmentTransaction. hide()：隐藏当前的 Fragment，仅仅是设为不可见，并不会销毁。

（5）FragmentTransaction. show()：显示之前隐藏的 Fragment。

（6）FragmentTransaction. detach()：会将 View 从 UI 中移除，和 remove()不同，此时 Fragment 的状态依然由 FragmentManager 维护。

（7）FragmentTransaction. attach()：重建 View 视图，附加到 UI 上并显示。

（8）FragmentTransaction. commit()：提交一个事务。在一个事务开启到提交可以进行多个 Fragment 的添加、移除和替换等操作。需要注意，FragmentTransaction 的 commit()方法一定要在 Activity. onSaveInstance()方法之前调用。

4. Fragment 的子类

Fragment 有以下子类，可实现不同类型的 UI 面板。

（1）DialogFragment 类：显示一个浮动的对话框。

（2）ListFragment 类：显示一个由 Adapter 管理项目的列表，类似于 ListActivity，它提供一些方法来管理一个 ListView，使用 onListItemClick 回调来处理单击事件。

（3）PreferenceFragment 类：显示一个 Preference 对象的层次结构的列表，类似于 PreferenceActivity。

Android 引入 Fragment 的初衷是为了适应大屏幕的平板电脑，下面的例题介绍使用 Fragment 模拟平板电脑的显示。

【例 3. 2】 模拟平板电脑划分为左右两个片段，分别显示 Java 概念列表和定义。

【解题思路】

在 Activity 界面左部的 Fragment 显示 Java 概念列表项，当单击某一列表项时，右部 Fragment 进行动态更新，显示对应的 Java 概念定义。

【开发步骤和程序分析】

(1) 在 Eclipse 中创建一个 ActivityFragment 应用项目，包名为 com. application . activityfragment。

(2) 设计布局。

布局文件为 activitytwopanes. xml 和 fragmentdetail. xml。

在 res/layout 目录下的 activitytwopanes. xml 文件中，左部添加一个 Fragment 元素，右部添加一个 FrameLayout 容器。

该文件编辑代码如下。

```
<?xml version = "1.0" encoding = "utf - 8"?>

<! -- 定义一个水平的 LinearLayout,并指定使用中等分隔条 -->
<LinearLayout
    xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = "horizontal"
    android:layout_width = "match_parent"
    android:layout_height = "match_parent"
    android:layout_marginLeft = "16dp"
    android:layout_marginRight = "16dp"
    android:divider = "?android:attr/dividerHorizontal"
    android:showDividers = "middle">

    <! -- 添加一个 Fragment,位于左部 -->
    <fragment
        android:name = "com.application.activityfragment.FragmentListConcept"
        android:id = "@ + id/concept_list "
        android:layout_width = "0dp"
        android:layout_height = "match_parent"
        android:layout_weight = "1" />

    <! -- 添加一个 FrameLayout 容器,位于右部 -->
    <FrameLayout
        android:id = "@ + id/concept_detail_container"
        android:layout_width = "0dp"
        android:layout_height = "match_parent"
        android:layout_weight = "3" />
</LinearLayout>
```

在 res/layout 目录下的 fragmentdetail. xml 文件中包含两个文本框，上边的文本框用于显示概念名称，下边的文本框用于显示概念内容。

该文件编辑代码如下：

```
<?xml version = "1.0" encoding = "utf - 8"?>
<! -- 定义一个垂直分布的线性布局 -->
< LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:layout_width = "match_parent"
    android:layout_height = "match_parent"
    android:orientation = "vertical">
    <! -- 定义一个 TextView 来显示概念名称,位于上边 -->
    < TextView
        style = "?android:attr/textAppearanceLarge"
        android:id = "@ + id/concept_title"
        android:layout_width = "match_parent"
        android:layout_height = "wrap_content"
        android:padding = "16dp"/>

    <! -- 定义一个 TextView 来显示概念内容,位于下边 -->
    < TextView
        style = "?android:attr/textAppearanceMedium"
        android:id = "@ + id/concept_desc"
        android:layout_width = "match_parent"
        android:layout_height = "match_parent"
        android:padding = "16dp"/>
</LinearLayout >
```

（3）在 com.application.activityfragment 包下的 ActivityFragmentConcept.java 文件中，加载 activitytwopane.xml 布局文件并实现 Callbacks 接口，该 Activity 左部的 Fragment 显示 Java 概念列表项，右部 Fragment 显示 Java 概念定义并进行动态更新。

该文件编辑代码如下：

```
package com.application.activityfragment;

import com.application.activityfragment.R;
import android.app.Activity;
mport android.os.Bundle;

//定义一个类 ActivityFragmentConcept 继承 Activity 类, 且实现 Callbacks 接口
public class ActivityFragmentConcept extends Activity implements
            FragmentListConcept.Callbacks
    {

        //重写 onCreate()方法
        @Override
        public void onCreate(Bundle savedInstanceState)
        {
            super.onCreate(savedInstanceState);
            //加载/res/layout 目录下的 activitytwopanes.xml 布局文件
```

```
18            setContentView(R.layout.activitytwopanes);
19        }
20        //重写 onItemSelected()方法，实现 Callbacks 接口必须实现的方法
21        @Override
22        public void onItemSelected(Integer id)
23        {
24            //创建 Bundle,准备向 Fragment 传入参数
25            Bundle arguments = new Bundle();
26            arguments.putInt(FragmentDetailConcept.ITEM_ID, id);
27            //创建 FragmentDetailConcept 对象
28            FragmentDetailConcept fragment = new FragmentDetailConcept();
29            //向 Fragment 传入参数
30            fragment.setArguments(arguments);
31            //使用 fragment 替换 concept_detail_container 容器当前显示的 Fragment
32            getFragmentManager().beginTransaction()
33                .replace(R.id.concept_detail_container, fragment)
34                .commit();
35        }
36 }
```

① 第 8 行至第 36 行定义一个类 ActivityFragmentConcept 继承 Activity 类，且实现 Callbacks 接口。

② 第 13 行至第 19 行重写 onCreate()方法，第 18 行加载/res/layout 目录下的 activitytwopanes.xml 布局文件。

③ 第 21 行至第 35 行重写 onItemSelected()方法，这是实现 Callbacks 接口必须实现的方法，第 25 行至第 26 行创建 Bundle，准备向 Fragment 传入参数，第 28 行为创建 FragmentDetailConcept 对象 fragment，第 30 行向 Fragment 传入参数，第 32 行至第 34 行使用 fragment 替换 concept_detail_container 容器当前显示的 Fragment。

(4) 下面的 Fragment 将会加载 fragmentdetail.xml 布局文件，构成 Activity 界面的右部，并根据传入的参数更新该部分。

在 com.application.activityfragment 包下的 FragmentDetailConcept.java 文件中，编辑代码如下：

```
package com.application.activityfragment;

import com.application.activityfragment.R;
import com.application.activityfragment.model.ConceptData;
import android.app.Fragment;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
    import android.widget.TextView;

```

```
12 //定义一个类 FragmentDetailConcept 继承 Fragment 类
13 public class FragmentDetailConcept extends Fragment
14 {
15     public static final String ITEM_ID = "item_id";
16     //保存该 Fragment 显示的 Concept 对象
17     ConceptData.Concept concept;
18
19     //重写 onCreate()方法
20     @Override
21     public void onCreate(Bundle savedInstanceState)
22     {
23         super.onCreate(savedInstanceState);
24         //如果启动该 Fragment 时包含了 ITEM_ID 参数
25         if (getArguments().containsKey(ITEM_ID))
26         {
27             concept = ConceptData.ITEM_MAP.get(getArguments()
28                 .getInt(ITEM_ID));
29         }
30     }
31
32     //重写 onCreateView()方法,该方法返回的 View 将作为 Fragment 显示的组件
33     @Override
34     public View onCreateView(LayoutInflater inflater,
35         ViewGroup container, Bundle savedInstanceState)
36     {
37         //加载/res/layout/目录下的 fragmentdetail.xml 布局文件
38         View rootView = inflater.inflate(R.layout.fragmentdetail,
39                 container, false);
40         if (concept != null)
41         {
42             //让 concept_title 文本框显示 concept 对象的 title 属性
43             ((TextView) rootView.findViewById(R.id.concept_title))
44                     .setText(concept.title);
45             //让 concept_desc 文本框显示 concept 对象的 desc 属性
46             ((TextView) rootView.findViewById(R.id.concept_desc))
47                 .setText(concept.desc);
48         }
49         return rootView;
50     }
51 }
```

① 第 13 行至第 51 行定义一个类 FragmentDetailConcept 继承 Fragment 类。

② 第 20 行至第 30 行重写 onCreate()方法。

③ 第 33 行至第 50 行重写 onCreateView()方法,该方法返回的 View 将作为 Fragment 显示的组件,第 38 行至第 39 行加载/res/layout/目录下的 fragmentdetail.xml 布局文件,

构成 Activity 界面的右部，第 43 行至第 44 行让 concept_title 文本框显示 concept 对象的 title 属性，第 46 行至第 47 行让 concept_desc 文本框显示 concept 对象的 desc 属性。

(5) 下面的 Fragment 开发了一个 ListFragment 的子类，调用 setListAdapter()方法设置 Adapter，构成 Activity 界面的左部的列表项。

在 com.application.activityfragment 包下的 FragmentListConcept.java 文件中，编辑代码如下：

```
package com.application.activityfragment;

import com.application.activityfragment.model.ConceptData;
import android.app.Activity;
import android.app.ListFragment;
import android.os.Bundle;
import android.view.View;
import android.widget.ArrayAdapter;
import android.widget.ListView;

    //定义一个类 FragmentListConcept 继承 ListFragment 类
    public class FragmentListConcept extends ListFragment
    {
        private Callbacks mCallbacks;
        //定义一个回调接口 Callbacks,该 Fragment 所在 Activity 需要实现该接口
        //该 Fragment 将通过该接口与它所在的 Activity 交互
        public interface Callbacks
        {
            public void onItemSelected(Integer id);
        }

        //重写 onCreate()方法
        @Override
        public void onCreate(Bundle savedInstanceState)
        {
            super.onCreate(savedInstanceState);
            //为该 ListFragment 设置 Adapter
            setListAdapter(new ArrayAdapter<ConceptData.Concept>(getActivity(),
                    android.R.layout.simple_list_item_activated_1,
                    android.R.id.text1, ConceptData.ITEMS));
        }

        //重写 onAttach()方法,当该 Fragment 被添加、显示到 Activity 时,回调该方法
        @Override
        public void onAttach(Activity activity)
        {
            super.onAttach(activity);
```

```
38              //如果 Activity 没有实现 Callbacks 接口,抛出异常
39              if (!(activity instanceof Callbacks))
40              {
41                  throw new IllegalStateException(
42                      BookListFragment 所在的 Activity 必须实现 Callbacks 接口!);
43              }
44              //把该 Activity 当成 Callbacks 对象
45              mCallbacks = (Callbacks)activity;
46          }
47          //重写 onDetach()方法,当该 Fragment 从它所属的 Activity 中被删除时回调该方法
48          @Override
49          public void onDetach()
50          {
51              super.onDetach();
52              //将 mCallbacks 赋为 null
53              mCallbacks = null;
54          }
55          //重写 onListItemClick()方法,当用户单击某列表项时激发该回调方法
56          @Override
57          public void onListItemClick(ListView listView
58              , View view, int position, long id)
59          {
60              super.onListItemClick(listView, view, position, id);
61              //激发 mCallbacks 的 onItemSelected 方法
62              mCallbacks.onItemSelected(ConceptData
63                  .ITEMS.get(position).id);
64          }
65
66          public void setActivateOnItemClick(boolean activateOnItemClick)
67          {
68              getListView().setChoiceMode(
69                      activateOnItemClick ? ListView.CHOICE_MODE_SINGLE
70                              : ListView.CHOICE_MODE_NONE);
71          }
72      }
```

① 第 12 行至第 72 行定义一个类 FragmentListConcept 继承 ListFragment 类。

② 第 17 行至第 20 行重写 onCreate()方法,定义一个回调接口 Callbacks,该 Fragment 所在 Activity 需要实现该接口,该 Fragment 将通过该接口与它所在的 Activity 交互。

③ 第 23 行至第 31 行重写 onCreate()方法,第 28 行至第 30 行为该 ListFragment 设置 Adapter。

④ 第 34 行至第 46 行重写 onAttach()方法,重写 onDetach()方法,当该 Fragment 从它所属的 Activity 中被删除时回调该方法。

⑤ 第 56 行至第 64 行重写 onListItemClick()方法,当用户单击某列表项时激发该回调方法。

（6）下面的 ConceptData 类使用 List 集合和 Map 集合记录系统所包含的 Concept 对象。

在 com. application. activityfragment. model 包下的 ConceptData. java 文件中，编辑代码如下：

```
package com.application.activityfragment.model;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

// 定义一个类 ConceptData
public class ConceptData
  {
      //定义一个内部类 Concept,作为系统的业务对象
      public static class Concept
      {

          public Integer id;
          public String title;
          public String desc;

          public Concept(Integer id, String title, String desc)
          {
              this.id = id;
              this.title = title;
              this.desc = desc;
          }

          @Override
          public String toString()
          {
              return title;
          }
      }
      //使用 List 集合记录系统所包含的 Concept 对象
      public static List<Concept> ITEMS = new ArrayList<Concept>();
      //使用 Map 集合记录系统所包含的 Concept 对象
      public static Map<Integer, Concept> ITEM_MAP
          = new HashMap<Integer, Concept>();

      static
      {
          //使用静态初始化代码,将 Concept 对象添加到 List 集合、Map 集合中
          addItem(new Concept(1, "类(Class)",
```

```
42                      "将数据和方法封装在一起的数据结构,用户定义一个类"
43                              + "实际上是定义一个新的数据类型."));
44              addItem(new Concept(2, "对象(Object)",
45                      "对象是类的实例,类是对象的模板."));
46              addItem(new Concept(3, "继承(Inheritance)",
47                      "继承可以实现代码的复用,被继承的类称为父类、基类或超类,"
48                              +"由继承而得的类称为子类或导出类.
49                                      子类继承父类的成员变量和成员方法,"
50                              + "可以修改父类的成员变量或重写父类的方法,"
51                              + "还可以添加新的成员变量和成员方法."));
52          }
53
54          private static void addItem(Concept concept)
55          {
56              ITEMS.add(concept);
57              ITEM_MAP.put(concept.id, concept);
58          }
59      }
```

① 第 9 行至第 59 行定义一个类 ConceptData。

② 第 33 行使用 List 集合记录系统所包含的 Concept 对象,第 35 行使用 Map 集合记录系统所包含的 Concept 对象。

③ 第 41 行至第 51 行使用静态初始化代码,将 Concept 对象添加到 List 集合、Map 集合中。

【运行结果】

在 Eclipse 中启动模拟器,然后运行项目 ActivityFragment,当左部 Fragment 的“继承(Inheritance)”列表项被选中,右部 Fragment 出现该条目的名称和内容,如图 3.7 所示。



图 3.7　ActivityFragment 运行结果

3.5 Intent 属性、过滤器和传递数据

无论是启动 Activity、启动 Service 或者启动 BroadcastReceiver 等某个组件，Android 使用统一的 Intent 来封装对某个组件的“启动意图”，以利于高层次的解耦。此外，Intent 还是组件之间通信的重要媒介。

3.5.1 Intent 属性

Intent 是连接应用程序的三个核心组件——Activity、Service 和 BroadcastReceiver 的桥梁，Intent 负责对应用中操作的动作、动作涉及数据及附加数据进行描述。

Intent 类定义在 android.content.Intent 包中，Intent 对象包含 Component、Action、Data、Category、Extra 及 Flag 等 6 种属性。

1. Component

Component(组件)属性用于指定 Intent 的目标组件，一般由相应组件的包名与类名组合而成。指定了 Component 属性值之后，Intent 的其他属性值都是可选的，此时该 Intent 就是一个显式 Intent。如果不指定 Component 属性值，则在 Androidmanifest 中，通过使用 IntentFilter 来找到一个与之匹配的目标组件，则该 Intent 就是个隐式 Intent。

通过 setComponent()方法设置组件属性，通过 setClass()、setClassName()方法设置将要启动的组件对应的类，通过 getComponent()方法读取组件名。

2. Action

Action(行动)属性用来指明要实施的动作是什么，其属性值是 Intent 即将触发动作名称的字符串。

Intent 定义了用大写字母和下画线组成的动作常量，如表 3.1 所示。

表 3.1　常用动作常量

常　量	含　义
ACTION_MAIN	应用程序入口
ACTION_VIEW	显示指定数据
ACTION_EDIT	编辑指定数据
ACTION_PICK	从列表中选择某项，并返回所选数据
ACTION_CALL	向指定用户打电话
ACTION_BATTERY_LOW	提示电池电量低
ACTION_SCREEN_ON	屏幕已开启

Action 属性可通过 setAction()方法来设置，通过 getAction()方法来读取。

3. Data

Data(数据)属性用于完成对 Intent 消息中数据的封装，描述 Intent 动作所操作数据的 URI(Uniform Resource Identifier，通用资源标识符)及 MIME(多用途互联网邮件扩展)。

Type(数据类型)属性用于指定 URI 对应的 MIME。

URI 格式为：scheme://host:port/path。

获取一个 URI 的语句格式为：

```
Uri uri = Uri.parse(<字符串>);
```

创建一个 Intent 对象的语句格式为：

```
Intent intent = new Intent(<动作>, <内容>);
```

代码：

```
Uri uri1 = Uri.parse(content://contacts/1);
Intent intent = new Intent(Intent.ACTION_VIEW, uri1);
```

说明：

在上面的代码中，uri1 是一个 Uri 变量，其值为：content://contacts/1，指向手机联系人信息集中的第一个联系人，创建的对象 intent 显示标识符为"1"的联系人的详细信息。

通过 setData()方法设置 URI，通过 getData ()方法读取 URI。

4. Category

Category(类别)属性用于描述目标组件额外的附加类别信息，其属性值是一个字符串。

一个 Intent 中可以包含多个 Category。如果没有设置 Category 属性值，Intent 会与在 Intent filter 中包含"android. category. DEFAULT"的 Activity 匹配。

Intent 定义了类别常量，如表 3.2 所示。

表 3.2 常用类别常量

常　　量	含　　义
CATEGORY_DEFAULT	默认的 Category
CATEGORY _BROWSABLE	指定该 Activity 能被浏览器安全调用
CATEGORY _HOME	设置该 Activity 随系统启动而运行
CATEGORY _LAUNCHER	该 Activity 列在应用程序启动器顶层，应用程序启动时首先被显示
CATEGORY _PREFERENCE	该 Activity 是参数面板
CATEGORY _INFO	用于提供包信息
CATEGORY _TEST	该 Activity 是一个测试

通过 addCategory()方法添加一个 Category，通过 emoveCategory()方法删除一个 Category，通过 getCategories()可以获取当前对象的所有 Category。

5. Extra

Extra(附加信息)属性用于在多个 Action 之间进行数据交换。

Extra 可以被当作一个 Bundle 对象，存入多组 key-value 对(键-值对)，这就可以通过 Intent 在不同 Action 之间进行数据交换了。

Intent 通过调用 putExtras()方法来添加一个新的键-值对，而在目标 Activity 中调用 getExtras()方法来获取 Extra 属性值。

6. Flag

Flag(标志)属性是一些有关系统如何启动组件的标志。指导 Android 系统启动一个 Activity 以及 Activity 启动后对其进行处理。

3.5.2 启动 Activity

启动 Activity 分为显式启动和隐式启动两种。显式启动，必须在 Intent 中指明启动的 Activity 对应的类。隐式启动不指明启动的 Activity 对应的类，系统会根据 Intent 指定的规则去启动符合条件的 Activity。

1. 显式启动

使用 Intent 显式启动 Activity，在创建一个 Intent 后，指定当前的应用程序上下文以及要启动的 Activity，把创建好的这个 Intent 作为参数传递给 startActivity()方法。

【例 3.3】 显式启动 Activity 示例。

【解题思路】

在应用项目 ExplicitStart 中，包含两个 Activity，一个是 ExplicitStartActivity，另一个是 SecondActivity。

程序默认启动的 Activity 是 ExplicitStartActivity，进入 ExplicitStartActivity 界面，当用户单击“启动 Activity”按钮后，程序使用 Intent 显式启动的 Activity 是 SecondActivity，显式启动 Activity 的代码如下：

```
Intent intent = new Intent(ExplicitStartActivity.this, SecondActivity.class);
startActivity(intent);
```

【开发步骤和程序分析】

(1) 在 Eclipse 中创建一个 ExplicitStart 应用项目，包名为 com.application.explicitstart。

(2) 在 src/com.application.explicitstart 包下的 ExplicitStartActivity.java 文件中，使用 Intent 显式启动 SecondActivity。

在该文件中编辑代码如下：

```
package com.application.explicitstart;

import com.application.explicitstart.R;
import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;

public class ExplicitStartActivity extends Activity {

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        Button button = (Button)findViewById(R.id.btn);
        button.setOnClickListener(new OnClickListener(){
```

```
19              public void onClick(View view){
20                  Intent intent = new Intent(ExplicitStartActivity.this, SecondActivity.class);
21                  startActivity(intent);
22              }
23          });
24      }
25  }
```

第 19 行至第 21 行(加黑部分),在单击事件 onClick(View view)方法中,首先,使用 Intent 构造方法创建一个实例 intent,其中的第一个参数是应用程序上下文 ExplicitStartActivity,第 2 个参数是接收 Intent 的目标组件 SecondActivity,这里使用显式启动方式,直接指明了需要启动的 Activity,然后,使用 startActivity(intent)方法显式启动 SecondActivity。

【运行结果】

应用项目 ExplicitStart 运行结果如图 3.8 和图 3.9 所示。

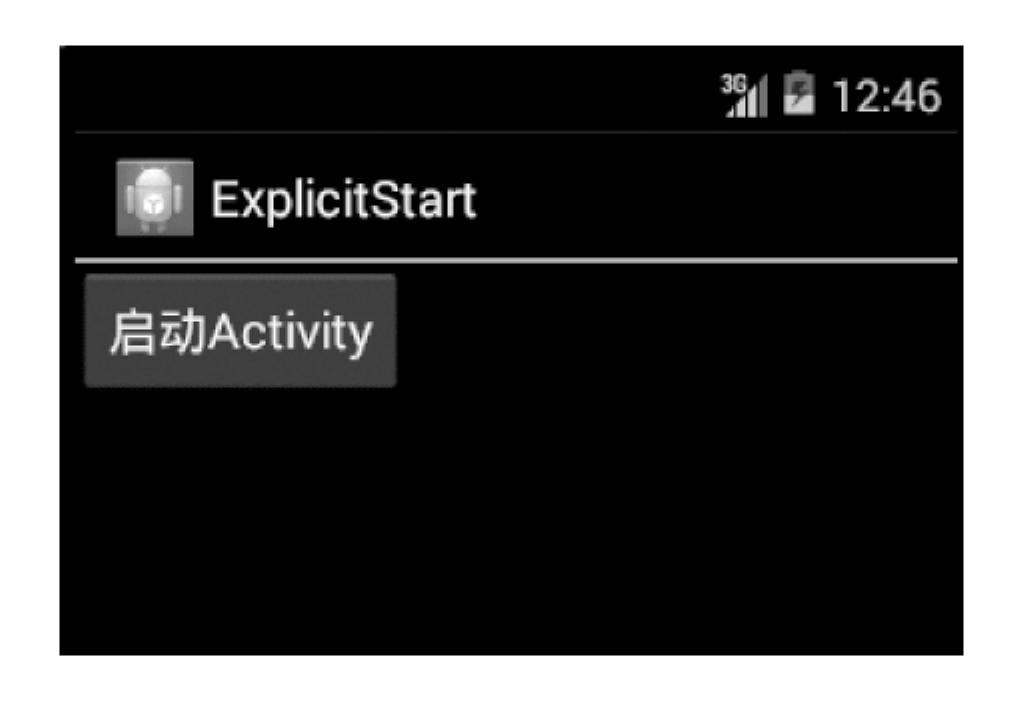




图 3.8 默认启动 ExplicitStartActivity



图 3.9 显式启动 SecondActivity

2. 隐式启动

隐式启动不指明启动的 Activity 对应的类,Android 系统会根据 Intent 指定的属性:Action、Data、Category 去启动符合条件的 Activity。

隐式启动的优点是不必指明需要启动哪一个 Activity,而由系统来决定,这样有利于降低组件之间的耦合度,提高 Android 组件的可复用性。

【例 3.4】 隐式启动 Activity 示例。

【解题思路】

在应用项目 ImplicitStart 中,需要启动网页 http://www.baidu.com。

在隐式启动 Activity 中,Intent 的动作是 Intent.ACTION_VIEW,数据是 Web 地址,使用 Uri.parse(urlString)方法。Android 系统在匹配 Intent 时,根据动作 Intent.ACTION_VIEW 和数据提供的是 Web 地址 http://www.baidu.com,判定 Intent 需要启动具有网页浏览功能的 Activity。

隐式启动 Activity 的代码如下:

```
Intent intent = new Intent(Intent.ACTION_VIEW, Uri.parse(urlString));
```

```
startActivity(intent);
```

【开发步骤和程序分析】

(1) 在 Eclipse 中创建一个 ImplicitStart 应用项目,包名为 com.application.implicitstart。

(2) 在 src/com.application.implicitstart 包下的 ImplicitStart.java 文件中,使用 Intent 隐式启动 Activity,提示用户在 http://后输入 Web 地址,以启动具有网页浏览功能的 Activity。

在该文件中编辑代码如下:

```
1  package com.application.implicitstart;
2
3  import com.application.implicitstart.R;
4  import android.app.Activity;
5  import android.content.Intent;
6  import android.net.Uri;
7  import android.os.Bundle;
8  import android.view.View;
9  import android.view.View.OnClickListener;
10 import android.widget.Button;
11 import android.widget.EditText;
12
13 public class ImplicitStart extends Activity {
14
15     @Override
16     public void onCreate(Bundle savedInstanceState) {
17         super.onCreate(savedInstanceState);
18         setContentView(R.layout.main);
19         final EditText editText = (EditText)findViewById(R.id.edit_url);
20         final Button button = (Button)findViewById(R.id.btn);
21         button.setOnClickListener(new OnClickListener(){
22             public void onClick(View view){
23                String urlString = editText.getText().toString();
24                Intent intent = new Intent(Intent.ACTION_VIEW, Uri.parse(urlString));
25                startActivity(intent);
26             }
27         });
28     }
29 }
```

第 22 行至第 25 行(加黑部分),在单击事件 onClick(View view)方法中,首先,使用 Intent 构造方法创建一个实例 intent,其中第一个参数的动作是显示数据 Intent.ACTION_VIEW,第 2 个参数的数据是 Web 地址,使用 Uri.parse(urlString)方法,这里使用隐式启动方式,提示用户在 http://后输入 Web 地址,当用户输入完成 Web 地址 http://www.baidu.com 并单击“浏览网页”按钮后,启动具有网页浏览功能的 Activity,显示百度页面。

【运行结果】

应用项目 ExplicitStart 运行结果如图 3.10 和图 3.11 所示。





图 3.10　提示用户在 http://后输入 Web 地址



图 3.11　隐式启动 Web 页面

3.5.3　Intent 过滤器

Intent 过滤器(Intent Filter)是一个包含 Intent 对象的 action、data、category 属性限制条件的集合，Intent Filter 要检测隐式 Intent 的 action、data、category 这三个属性，其中任何一项失败，Android 系统都不会传递 Intent 给此组件。

一个组件可以有多个 Intent Filter，Intent 只要通过其中的某个 Intent Filter 检测，就可以调用此组件。

Intent 过滤器在 AndroidManifest.xml 文件中进行声明，Intent 过滤器使用< intent-filter >子标签来进行声明。

Intent 解析机制主要是通过查找已注册在 AndroidManifest.xml 中的所有 Intent Filter 及其中定义的 Intent 属性，最终找到匹配的 Intent。

(1) Action 检查：一个 Intent 只能设置一种 Action，而一个 Intent Filter 可以设置多个 Action。如果 Intent 指明定了 action，则目标组件的 Intent Filter 的 action 列表中就必须包含有这个 action，否则不能匹配；如果 Intent 没有指定 action，将自动通过检查。

(2) Category 检查：在一个 Intent Filter 中，可以设置多个 Category。如果 Intent 指定了一个或多个 category，这些类别必须全部出现在组件的 category 列表中。

(3) Data 检查：对数据的检查有两部分，一是对数据 URI 进行检查，一是对数据类型

进行检查，对数据 URI 的检查包括 schema、authority 和 path。

【例 3.5】 Intent 过滤器示例。

【解题思路】

在应用项目 IntentFilterExample 中，在 AndroidManifest.xml 文件<intent-filter>节点的<action>标签、<category>标签和<data>标签，分别定义 Intent 过滤器的“动作”“类别”和“数据”，以进行相关检查，最终找到匹配的 Intent。

【开发步骤和程序分析】

(1) 在 Eclipse 中创建一个 IntentFilterExample 应用项目，包名为 com.application.IntentFilterExample。

(2) 在 AndroidManifest.xml 文件中，分别定义了 UserActivity1 和 UserActivity2 的 Intent 过滤器，包括动作、类别和数据等。

在该文件中编辑代码如下：

```
1   <?xml version = "1.0" encoding = "utf - 8"?>
2   < manifest xmlns:android = "http://schemas.android.com/apk/res/android"
3       package = "com.application.IntentFilterExample"
4       android:versionCode = "1"
5       android:versionName = "1.0" >
6       < uses - sdk android:minSdkVersion = "14" />
7       < application
8           android:icon = "@drawable/ic_launcher"
9           android:label = "@string/app_name" >
10          < activity
11              android:label = "@string/app_name"
12              android:name = "com.application.IntentFilterExample.UserActivity1" >
13              < intent - filter >
14                  < action android:name = "android.intent.action.MAIN" />
15                  < category android:name = "android.intent.category.LAUNCHER" />
16              </intent - filter >
17          </activity >
18          < activity android:name = "com.application.IntentFilterExample.UserActivity2"
19                  android:label = "@string/app_name">
20              < intent - filter >
21                 < action android:name = "android.intent.action.VIEW" />
22                 < category android:name = "android.intent.category.DEFAULT" />
23                 < data android:scheme = "schemodemo" android:host = "com.application" />
24              </intent - filter >
25          </activity >
26      </application >
```

① 在第 10 行到第 17 行，定义了第 1 个 Activity 及其 Intent 过滤器，第 1 个 Activity 名为 UserActivity1，第 13 行到第 16 行是第 1 个 Activity 的 Intent 过滤器(加黑部分)，动作为 android.intent.action.MAIN，类别为 android.intent.category.LAUNCHER，由此得

出，第 1 个 Activity 是应用程序启动后显示的默认用户界面。

② 在第 18 行到第 25 行，定义了 2 个 Activity 及其 Intent 过滤器，第 2 个 Activity 名为 UserActivity2，第 20 行到第 24 行是第 2 个 Activity 的 Intent 过滤器（加黑部分），过滤器的动作是 android. intent. action. VIEW，表示根据 Uri 协议，以浏览的方式启动相应的 Activity；类别是 android. intent. category. DEFAULT，表示数据的默认动作；数据的协议部分是 android: scheme＝"schemodemo"，数据的主机名称部分是 android: host＝"com. application"。

（3）在 src/com. application. IntentFilterExample 包下的 UserActivity1. java 文件中，定义的 Intent 动作和数据分别与 UserActivity2 的 Intent 过滤器定义的动作、数据要求相匹配，该 Intent 用于启动 UserActivity2。

在该文件中编辑代码如下：

```
1  package com.application.IntentFilterExample;
2
3  import com.application.IntentFilterExample.R;
4  import android.app.Activity;
5  import android.content.Intent;
6  import android.net.Uri;
7  import android.os.Bundle;
8  import android.view.View;
9  import android.view.View.OnClickListener;
10 import android.widget.Button;
11
12 public class UserActivity1 extends Activity {
13
14     @Override
15     public void onCreate(Bundle savedInstanceState) {
16         super.onCreate(savedInstanceState);
17         setContentView(R.layout.main);
18         Button button = (Button)findViewById(R.id.btn);
19         button.setOnClickListener(new OnClickListener(){
20             public void onClick(View view){
21                 Intent intent = new Intent(Intent.ACTION_VIEW, Uri.parse("schemodemo://
                   com.application/path"));
22                 startActivity(intent);
23             }
24         });
25     }
26 }
```

① 第 21 行至第 22 行，定义了一个 Intent 用来启动另一个 Activity，这个 Intent 与 Activity 设置的 Intent 过滤器是完全匹配的。

② 在第 21 行定义的 Intent（加黑部分），动作为 Intent. ACTION_VIEW，Uri 是 "schemodemo://edu. hrbeu/path"，其中的协议部分为"schemodemo"，主机名部分为"edu.

hrbeu”,分别与 Intent 过滤器定义的动作、数据要求完全匹配。因此,当代码第 18 行定义的 Intent,在 Android 系统与 Intent 过滤器列表进行匹配时,会与 AndroidManifest. xml 文件中 UserActivity2 定义的 Intent 过滤器完全匹配。

【运行结果】

应用项目 IntentFilterExample 运行结果如图 3.12 和图 3.13 所示。



图 3.12　进入 UserActivity1 界面



图 3.13　进入 UserActivity2 界面

3.5.4　Activity 组件之间通过 Intent 通信

下面通过一个例题说明 Activity 组件之间通过 Intent 通信。

【例 3.6】　Activity 组件之间通过 Intent 通信举例。

两个 Activity:FirstActivity 和 SecondActivity,界面上首次进入的 Activity 为 FirstActivity,通过单击按钮来实现 FirstActivity 和 SecondActivity 的相互跳转,使用 Intent 对象实现两个 Activity 之间的通信。

【解题思路】

在应用项目 ActivityIntentExample 中,FirstActivity 的布局文件为 first_main. xml,SecondActivity 的布局文件为 second_main. xml,FirstActivity 的 Java 代码文件为 FirstActivity. java,SecondActivity 的 Java 代码文件为 SecondActivity. java。

在 FirstActivity 和 SecondActivity 组件中,使用了按钮控件,因此在相应的布局文件中,需要声明按钮控件,其标签为<Button>。

在 Java 代码文件中,对按钮控件设置监听,使用方法 setOnClickListener(),如果监听到按钮被单击,则执行 onClick()事件方法定义的操作。

在两个 Activity 调用中,使用显式启动,两个 Activity 调用需要返回信息,使用 startActivityForResult()方法发送 Intent 对象,startActivityForResult()方法的格式如下:

```
startActivityForResult(Intent intent, int requestCode)
```

如果是从 A 发送 B,然后从 B 返回到 A,并且需要传递信息,则在 A 代码中使用 startActivityForResult()方法发送 Intent 到 B,并且重写 onActivityResult()方法用于处理返回的数据;在 B 代码中使用 setResult()方法准备好回传的数据,并且使用 finish()方法将打包好的数据发回给 A,并运行 A 中的 onActivityResult()部分代码。

【开发步骤和程序分析】

(1) 在 Eclipse 中创建一个 ActivityIntentExample 应用项目,包名为 com. application . activityintentexample,有两个 Activity:FirstActivity 和 SecondActivity。

(2) 设计布局。

布局文件为 first_main. xml 和 second_main. xml。

在 res/layout 目录中,编写 FirstActivity 的布局文件 first_main. xml,定义了一个"进入 SecondActivity"按钮。

在该文件中编辑代码如下:

```
1  <?xml version = "1.0" encoding = "utf - 8"?>
2  < LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
3      android:orientation = "vertical"
4      android:layout_width = "fill_parent"
5      android:layout_height = "fill_parent"
6      >
7    < Button android:id = "@ + id/button1"
8          android:layout_width = "wrap_content"
9          android:layout_height = "wrap_content"
10         android:text = "进入 SecondActivity" />
11 </LinearLayout >
```

第 7 行至第 10 行,定义了一个按钮,其中,第 7 行定义该按钮的 id 变量名为 button1,并添加到 R. java 文件中,为 Java 代码提供调用,第 10 行定义该按钮显示文本内容为"进入 SecondActivity"。

在 res/layout 目录中,编写 SecondActivity 的布局文件 second_main. xml,定义了一个"返回 FirstActivity"按钮。

在该文件中编辑代码如下:

```
1  <?xml version = "1.0" encoding = "utf - 8"?>
2  < LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
3      android:orientation = "vertical" android:layout_width = "fill_parent"
4      android:layout_height = "fill_parent">
5      < Button android:id = "@ + id/button2"
6          android:layout_width = "wrap_content"
7          android:layout_height = "wrap_content"
8          android:text = "返回 FirstActivity" />
9 </LinearLayout >
```

第 5 行至第 8 行,定义一个按钮,其中,第 5 行定义该按钮的 id 变量名为 button2,并添加到 R. java 文件中,第 8 行定义该按钮显示文本内容为"返回 FirstActivity"。

(3) 在包 com. application. activityintentexample 下的 FirstActivity. java 文件中,加载 first _ main. xml 布局文件,使用 startActivityForResult ( ) 方法发送 Intent1 到 SecondActivity,并重写 onActivityResult()方法用于处理返回的数据。

在该文件中编辑代码：

```java
package com.application.activityintentexample;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
public class FirstActivity extends Activity {
    OnClickListener listener1 = null;
    Button button1;
    static final int REQUEST_CODE = 1;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        listener1 = new OnClickListener() {
            public void onClick(View v) {
                //创建一个 Intent 对象,对象名为 intent1
                Intent intent1 = new Intent(FirstActivity.this, SecondActivity.class);
                //向 intent1 中添加附加信息,一组键-值对的名为"firstactivity",
                //值为"从 FirstActivity 进入--"
                intent1.putExtra("firstactivity", "从 FirstActivity 进入--");
                //使用 startActivityForResult()方法发送 intent1 对象,同时发送一个请求码
                startActivityForResult(intent1, REQUEST_CODE);
            }
        };
        setContentView(R.layout.first_main);
        button1 = (Button) findViewById(R.id.button1);
        button1.setOnClickListener(listener1);
        setTitle("查看信息内容页面        --首次进入 FirstActivity--");
  }
    //重写 onActivityResult()方法,通过判断请求码值和返回码值,来确定是否正确获得
    //回传数据,如果是正确的,则取出回传数据并显示在标题栏中
    @Override
    protected void onActivityResult(int requestCode, int resultCode, Intent data) {
        if (requestCode == REQUEST_CODE) {
            if (resultCode == RESULT_CANCELED)
                setTitle("取消");
            else if (resultCode == RESULT_OK) {
                String temp = null;
                 Bundle extras = data.getExtras();
                    if (extras != null) {
```

```
44                      temp = extras.getString("store");
45                  }
46                  setTitle("查看信息内容页面        --" + temp);
47              }
48          }
49      }
50 }
```

① 第 17 行至第 27 行(加黑部分),创建一个监听 listener1,同时定义一个 onClick 事件,在事件中定义了当监听到按钮被单击,则进行相应的操作。其中:

- 第 20 行创建一个 Intent 对象,对象名为 intent1,其动作值为 FirstActivity.this,数据值为 SecondActivity.class,这是使用 Intent 显式启动 Activity。
- 第 23 行向 intent1 中添加附加信息:一组键-值对的 Bundle 信息,其名为“firstactivity”,值为“从 FirstActivity 进入--”。
- 第 25 行使用 startActivityForResult()方法发送 intent1 对象,同时发送一个请求码 REQUEST_CODE 给 SecondActivity。

② 第 35 行至第 49 行(加黑部分)重写 onActivityResult()方法,通过判断请求码值和返回码值,来确定是否正确获得回传数据,如果是正确的,则取出回传数据并显示在标题栏中。

(4) 在包 com.application.activityintentexample 下的 SecondActivity.java 文件中,加载 second_main.xml 布局文件,使用 setResult()方法准备好回传的数据,并且使用 finish()方法将打包好的数据发回给 FirstActivity。

在该文件中编辑代码:

```
package com.application.activityintentexample;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;

public class SecondActivity extends Activity {
    OnClickListener listener1 = null;
    Button button1;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.second_main);
        listener1 = new OnClickListener() {
            public void onClick(View v) {
                //创建一个 Bundle 对象,对象名为 bundle
```

```
21                  Bundle bundle = new Bundle();
22                  //将一组键-值对保存到 bundle,其名为 "store",
23                  //其值为"自 SecondActivity 返回--"
24                  bundle.putString("store", "自 SecondActivity 返回--");
25                  //创建一个 Intent 对象,对象名为 intent2
26                  Intent intent2 = new Intent();
27                  //向 intent2 中添加已保存在 bundle 中的键-值对信息
28                  intent2.putExtras(bundle);
29                  //打包回传的数据,包括返回码值 RESULT_OK 和 intent2 对象
30                  setResult(RESULT_OK, intent2);
31                  //将打包好的数据发回给 FirstActivity,并且运行
32                  //FirstActivity.java 中 onActivityResult()里面的代码
33                  finish();
34              }
35          };
36          button1 = (Button) findViewById(R.id.button2);
37          button1.setOnClickListener(listener1);
38          //从 FirstActivity 的 intent1 对象中取出附加信息赋值给 extras,如果其键-值对非空,
39          //则取出名为"firstactivity"的对应值给字符串变量 data,并将 data 的值显示在标题栏中
40          String data = null;
41          Bundle extras = getIntent().getExtras();
42              if (extras != null) {
43                  data = extras.getString("firstactivity");
44              }
45          setTitle("显示信息内容页面          -- " + data);
46      }
47  }
```

① 第 18 行至第 35 行(加黑部分),创建一个监听 listener1,同时定义一个 onClick()方法,在方法中定义了当监听到按钮被单击,则进行相应的操作。其中:

- 第 21 行和第 24 行创建一个 Bundle 对象,对象名为 bundle,并将一组键-值对保存到其中,其名为"store",其值为"自 SecondActivity 返回--"。
- 第 26 行创建一个 Intent 对象,对象名为 intent2。
- 第 28 行向 intent2 中添加附加信息,此附加信息是已保存在 bundle 中的键-值对信息。
- 第 30 行是打包回传的数据,包括返回码值 RESULT_OK 和 intent2 对象。
- 第 33 行将打包好的数据发回给 FirstActivity,并且运行 FirstActivity.java 中 onActivityResult()里面的代码。

② 第 40 行至第 45 行(加黑部分)从 FirstActivity 的 intent1 对象中取出附加信息赋值给 extras,如果其键-值对非空,则取出名为"firstactivity"的对应值给字符串变量 data,并将 data 的值显示在标题栏中。

(5) 编写根目录下的 AndroidManifest.xml 文件的代码,增加 Activity 组件 SecondActivity 的声明。

```
<?xml version = "1.0" encoding = "utf - 8"?>
< manifest xmlns:android = "http://schemas.android.com/apk/res/android"
      package = "com.application.activityintentexample"
      android:versionCode = "1"
      android:versionName = "1.0" >
      < uses - sdk android:minSdkVersion = "10" />
      < application
          android:icon = "@drawable/ic_launcher"
          android:label = "@string/app_name" >
           < activity
               android:name = ".FirstActivity"
               android:label = "@string/app_name" >
               < intent - filter >
                   < action android:name = "android.intent.action.MAIN" />
                   < category android:name = "android.intent.category.LAUNCHER" />
              </intent - filter >
          </activity >
          //增加 Activity 组件 SecondActivity 的声明
          < activity android:name = ".SecondActivity"></activity >
      </application >
</manifest >
```

【运行结果】

应用项目 IntentFilterExample 运行结果如图 3.14～图 3.16 所示。



图 3.14　首次进入“查看信息内容页面”



图 3.15　单击“进入”按钮后进入“显示信息内容页面”



图 3.16　单击“返回”按钮后返回“查看信息内容页面”

3.6 小　　结

本章主要介绍了以下内容：

(1) Android 应用程序生命周期指从启动到终止的全过程，应用程序的生命周期是由 Android 系统进行调度和控制，而不是由应用程序直接控制的。Android 应用程序组件有其生命周期，指从创建到销毁的全过程，Activity 组件是 Android 应用生命周期的重要部分之一。

(2) 进程(Process)是程序的一次执行，进程由程序、数据和进程控制块构成，进程是一个可拥有资源的独立实体，又是一个可以独立调度的基本单位。在 Android 操作系统中，进程是应用程序的具体实现。组件运行的进程由 Androidmanifest 文件控制。

线程(Thread)是进程中的一个实体，是被系统独立调度的基本单位。线程基本上不拥有系统资源，只有一些在运行中必不可少的资源(如程序计数器、一组寄存器和栈)，但它可共享所属进程的全部资源。

(3) Android 应用中常用的基本组件有 Activity(活动)、Service(服务)、BroadcastReceiver(广播接收器)、ContentProvider(数据提供器)、Intent(意图)等。

Activity 用于提供可视化用户界面，它是最常用的组件；Service 是一个常用组件，一般用于没有用户界面，又需要长时间在后台运行的应用；BroadcastReceiver 是另一个常用组件，用来接收广播消息，不包含任何用户界面，其监听的事件源是其他组件；ContentProvider 组件是 Android 系统提供的一种标准的共享数据的机制，用来管理和共享应用程序的数据存储；Intent 是不同组件间通信的载体，是连接各个组件的桥梁。

(4) Activity 的生命周期中存在五种状态：启动状态，运行状态，暂停状态，停止状态，销毁状态。Activity 的回调方法有：onCreate()、onStart()、onResume()、onPause()、onStop()、onRestart()、onDestroy()、onSaveInstanceState()、onRestoreInstanceState()等。Activity 的生命周期可分为完全生命周期、可视生命周期和活动生命周期，每种生命周期中包含不同的回调方法。

(5) Fragment 有自己的生命周期，但它的生命周期受其所在的 Activity 生命周期控制，Fragment 不能独立存在，它必须嵌入到 Activity 中。当 Activity 暂停时，它拥有的所有的 Fragment 都暂停了；当 Activity 销毁时，它拥有的所有 Fragment 都被销毁；当 Activity 处于活动状态时(在 onResume()之后，onPause()之前)，用户可以通过方法操作每个 Fragment。

(6) 无论是启动 Activity、启动 Service 或者启动 BroadcastReceiver 等某个组件，Android 使用统一的 Intent 来封装对某个组件的"启动意图"，以利于高层次的解耦。此外，Intent 还是组件之间通信的重要媒介。Intent 对象包含 Component、Action、Data、Category、Extra 及 Flag 等 6 种属性。启动 Activity 分为显式启动和隐式启动两种。Intent 过滤器(Intent Filter)是一个包含 Intent 对象的 action、data、category 属性限制条件的集合，Intent Filter 要检测隐式 Intent 的 action、data、category 这三个属性，其中任何一项失

败，Android 系统都不会传递 Intent 给此组件。

习　题　3

一、选择题

3.1　在 Android 系统的进程优先级中，高优先级的进程是________。
A. 服务进程　　B. 可见进程　　C. 后台进程　　D. 前台进程

3.2　用于提供可视化用户界面的组件是________。
A. Service　　B. Activity
C. ContentProvider　　D. BroadcastReceiver

3.3　没有用户界面、长时间在后台运行的组件是________。
A. Service　　B. Activity
C. ContentProvider　　D. BroadcastReceiver

3.4　________组件用来封装对某个组件的“启动意图”。
A. Service　　B. Activity
C. ContentProvider　　D. Intent

3.5　Activity 可见，获得焦点，可与用户进行交互的状态是________。
A. 启动状态　　B. 停止状态　　C. 暂停状态　　D. 运行状态

3.6　Activity 失去焦点，但仍可见，依然保持活力的状态是________。
A. 启动状态　　B. 停止状态　　C. 暂停状态　　D. 运行状态

3.7　创建 Activity 时被回调，且只会被调用一次的方法是________。
A. onPause()　　B. onResume()　　C. onCreate()　　D. onStart()

3.8　恢复 Activity 时被回调，当 Activity 可以开始与用户进行交互之前被调用的方法是________。
A. onPause()　　B. onResume()　　C. onCreate()　　D. onStart()

3.9　显示指定数据的动作常量是________。
A. ACTION_MAIN　　B. ACTION_VIEW
C. ACTION_EDIT　　D. ACTION_CALL

3.10　应用程序入口的动作常量是________。
A. ACTION_MAIN　　B. ACTION_VIEW
C. ACTION_EDIT　　D. ACTION_CALL

二、填空题

3.11　Android 应用中常用的基本组件有 Activity、Service、BroadcastReceiver、ContentProvider 和________等。

3.12　Activity 的生命周期中存在五种状态：启动状态，暂停状态，停止状态，销毁状态和________。

3.13　Activity 的回调方法有：onCreate()、onResume()、onPause()、onStop()、onRestart()、onDestroy()和________。

3.14　Intent 对象包含 Component、Action、Data、Extra、Flag 和________ 6 种属性。

3.15　Intent 过滤器(Intent Filter)是一个包含 Intent 对象的 action、category 和________属性限制条件的集合。

三、问答题

3.16　简述 Activity 的生命周期中存在的五种状态和状态之间的转换关系。

3.17　简述 Android 应用中常用的基本组件及其用途。

3.18　简述 Activity 的回调方法和调用顺序。

3.19　简述 Android 系统的前台进程、可见进程、服务进程、后台进程、空进程以及前台进程的特点和优先级。

3.20　什么是 Fragment? Fragment 有哪些特点?

3.21　简述 Fragment 的生命周期。

3.22　Intent 对象包含哪几种属性? 各有何作用?

四、编程题

3.23　编写程序,使浏览器显示本校的主页。

3.24　编写使用 Intent 拨打电话的程序,并给你的同学打一个电话。

第 4 章　Android 基本控件

本章要点

- Android 应用开发的一个重要内容是用户界面开发，View 和 ViewGroup 是 Android 平台上最基本的两个用户界面表达单元。
- View（视图）是所有可视化窗体控件的基类，ViewGroup（视图组）可以充当 View 的容器。
- 常用的布局有线性布局（LinearLayout）、表格布局（TableLayout）、帧布局（FrameLayout）、网格布局（GridLayout）、相对布局（RelativeLayout）、绝对布局（AbsoluteLayout）。
- 常用的基本控件有 TextView、EditText、Button、ImageButton、ImageView、Checkbox、RadioButton、AnalogClock、DigitalClock、DatePicker、TimePicker 等。

用户界面（User Interface，UI）是系统和用户之间进行信息交换的媒介，控件是 Android 用户界面中的组成元素。本章介绍 Android 用户界面设计、常用布局、常用基本控件等内容。

4.1　用户界面设计

Android 应用开发的一个重要内容是用户界面开发，提供友好的用户界面是设计人员的主要工作。

1. 用户界面设计

用户界面（User Interface，UI）是系统和用户之间进行信息交换的媒介，实现信息的内部形式与人类可以接受形式之间的转换。当前流行的图形用户界面（Graphical User Interface，GUI）是采用图形方式与用户进行交互的界面。

Android 应用开发的一个重要内容是用户界面开发，一个应用项目如果没有提供友好的图形用户界面，再优秀的应用项目也很难吸引用户。与此相反，如果应用项目提供了友好的图形用户界面，往往会受到用户的欢迎。Android 提供了大量功能丰富的 UI 控件，界面设计人员按一定的规则采用“搭积木”的方式，并具备一定的设计经验后，就可设计出优秀的图形用户界面。

在 Android 手机上进行用户界面设计具有创造性和挑战性，由于界面设计与程序逻辑完全分离，手机界面设计人员与程序开发人员是独立并行工作的；而不同型号手机的尺寸、长宽比例和屏幕分辨率的不同，程序界面需要根据屏幕信息进行自动调整；并且需要设计人员合理利用有限的显示空间设计出优秀的手机界面。

2. View 类和 ViewGroup 类

View 和 ViewGroup 是 Android 平台上最基本的两个用户界面表达单元。

1) View 类

View(视图)是所有可视化窗体控件的基类,所有在界面上的可见元素都是 View 的子类。

控件是 Android 用户界面中的组成元素,控件的父类是 View。

View 对象是一个数据体,它的属性存储了屏幕上特定矩形区域的布局参数及内容。

每个 View 的子类对象都是 android. view. View 类的一个实例。

所有可视控件都继承 View 类属性,View 类常用的属性如表 4.1 所示。

表 4.1　View 类常用的属性

属　　性	说　　明
android:background	设置背景
android:clickable	设置 View 是否响应单击事件
android:visible	控制 View 的可见性
android:focusable	控件 View 是否可以获取焦点
android:id	为 View 设置标识符,可通过 findViewById 方法获取

2) ViewGroup 类

ViewGroup(视图组)是 android. view. Viewgroup 的一个实例,是一种特殊类型的视图,可以充当 View 的容器。

ViewGroup 的子控件既可以是 View 类,也可以是 ViewGroup 类。

使用 ViewGroup 可以创建比较复杂的界面元素。

4.2　常 用 布 局

Layout(布局)是 ViewGroup 类的子类,为视图控件提供排列结构。

Android 常用的布局方式有:线性布局(LinearLayout)、表格布局(TableLayout)、帧布局(FrameLayout)、网格布局(GridLayout)、相对布局(RelativeLayout)、绝对布局(AbsoluteLayout)。

4.2.1　定义布局文件和在 Activity 中引用布局文件

Android 中的布局管理一般在 XML 中进行规划和设计,其方法的优点是直观、简洁,实现了 UI 界面和 Java 逻辑代码的分离。

1. 定义布局文件

在资源目录 res\layout 中定义一个 XML 布局文件,例如 main. xml,在其中声明布局方式,其属性如下:

1) ID 属性

控件的 ID 属性用于在 Java 代码中引用这个相应的控件,当 Java 程序被编译时,这个

ID 作为一个整数被引用,但通常是在布局 XML 文件中以字符串的形式定义一个 ID:

```
android:id="@+id 字符串"
```

在定义资源之前一定要先使用 android:id 属性定义它的 ID 号,这个资源才能被记录到 R.java 中,然后才能在 Java 源代码中使用。

2) 尺寸参数

尺寸参数一般指 layout_height(布局高度)、layout_width(布局宽度)等参数。表示尺寸有采用参数和采用确定的数字两种方法。

(1) 采用参数。

- fill_parent

指定控件的高度、宽度与父容器的高度、宽度相同。

- match_parent

该属性值与 fill_parent 相同。

- wrap_content

指定控件的大小恰好能包裹它的内容。

(2) 采用确定的数字。

例如用确定的数字 30px 等。

3) xmlns:android 属性。

该属性一般取值为"http://schemas.android.com/apk/res/android"。这是定义了一个 XML 命名空间,告诉 Android 开发工具准备使用 android 命名空间里的一些通用属性。

2. 在 Activity 中引用布局文件

定义了 XML 布局文件后,在 src 目录中的 Activity 文件引用该布局文件。例如,在 src 目录中的 MainActivity.java 文件调用 setContentView()方法引用 main.xml 文件

```
setContentView(R.layout.main);
```

将布局中的控件实例化。

4.2.2 线性布局

线性布局(LinearLayout)是最常用的布局方式。线性布局的控件定义在 <LinearLayout></LinearLayout>标签之间,它将其包含的控件元素按水平或者垂直方向顺序排列。

线性布局可以嵌套:一个线性布局内可以再定义线性布局。

线性布局常用属性如表 4.2 所示。

表 4.2 线性布局常用属性

属　　性	说　　明
andriod:orientation	设置排列方向,horizontal 表示水平方向,vertical 表示垂直方向
android:layout_width	设置宽度,fill_parent 表示填充整个屏幕,wrap_content 表示按对象上文字宽度不同来确定显示对象宽度

续表

属　　性	说　　明
android:layout_height	设置高度,属性值同 android:layout_height
android:layout_weight	设置控件的重要程度,所有控件都有一个 weight 值,按比例为它们划分空间,默认为 0,数值越小越重要
andriod:gravity	设置内部元素的对齐方式,top 表示对齐到容器顶部,bottom 表示对齐到容器底部,left 表示对齐到容器左侧,right 表示对齐到容器右侧,enter 表示对齐到容器中央位置

【例 4.1】 创建水平线性布局和垂直线性布局。

【解题思路】

创建水平线性布局,有 4 个按钮,以横向线性排列方式显示排列在屏幕的顶部,需要设置线性布局的方向是 horizontal,即水平方向。

创建垂直线性布局,有 4 个按钮,以竖向线性排列方式显示排列在屏幕的右侧,需要设置线性布局的方向是 vertical,即垂直方向。

下面仅介绍创建水平线性布局的开发步骤和程序分析,创建垂直线性布局读者可参考以下步骤自行完成。

【开发步骤和程序分析】

(1) 在 Eclipse 中创建一个 LinearLayout 应用项目,包名为 com. application. linearlayout。

(2) 在 res/layout 目录下的 main. xml 文件中,声明了一个水平线性布局,横向线性排列 4 个按钮。

在该文件中编辑代码如下:

```
<?xml version = "1.0" encoding = "utf - 8"?>
<! -- 声明一个水平分布的线性布局 -->
<LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
                 android:layout_width = "fill_parent"
                 android:layout_height = "fill_parent"
                 android:orientation = "horizontal">
             <! -- 设置第 1 个按钮 -->
             <Button android:id = "@ + id/button1"
                     android:layout_width = "wrap_content"
                 android:layout_height = "wrap_content"
                 android:text = "Button1"
                 android:layout_weight = "1"
           />
          <! -- 设置第 2 个按钮 -->
           <Button android:id = "@ + id/button2"
                 android:layout_width = "wrap_content"
                 android:layout_height = "wrap_content"
                 android:text = "Button2"
                 android:layout_weight = "1"
```

```
20            />
21        <!-- 设置第 3 个按钮 -->
22         <Button android:id = "@ + id/button3"
23                 android:layout_width = "wrap_content"
24                 android:layout_height = "wrap_content"
25                 android:text = "Button3"
26                 android:layout_weight = "1"
27           />
28        <!-- 设置第 4 个按钮 -->
29         <Button android:id = "@ + id/button4"
30                 android:layout_width = "wrap_content"
31                 android:layout_height = "wrap_content"
32                 android:text = "Button4"
33                 android:layout_weight = "1"
34            />
35    </LinearLayout>
```

① 第 2 行至第 35 行声明了一个水平线性布局，其中，第 4 行至第 5 行设置线性布局的宽度和高度属性值为 fill_parent，即充满容器的宽和高，第 6 行设置线性布局的方向是 horizontal，即水平方向。

② 第 8 行至第 13 行，第 15 行至第 20 行，第 22 行至第 27 行，第 29 行至第 34 行，分别设置了 4 个按钮，第 9 行 wrap_content 表示用按钮文字宽度来确定按钮显示宽度。

(3) 在 src/ com. application. linearlayout 包下的 LinearLayoutActivity. java 文件中，加载 main. xml 布局文件，在该文件中编辑代码如下：

```
1  package com.application.linearlayout;
2
3  import com.application.linearlayout.R;
4  import android.app.Activity;
5  import android.os.Bundle;
6
7  public class LinearLayoutActivity extends Activity {
8
9      @Override
10     public void onCreate(Bundle savedInstanceState) {
11         super.onCreate(savedInstanceState);
12         //调用 setContentView()方法引用 main.xml 布局
13         setContentView(R.layout.main);
14     }
15 }
```

第 13 行调用 setContentView()方法引用 main. xml 文件，将布局中的控件实例化。

【运行结果】

在 Eclipse 中启动模拟器，然后运行项目 LinearLayout，水平线性布局运行结果如图 4. 1 所示。

垂直线性布局运行的结果如图 4.2 所示。

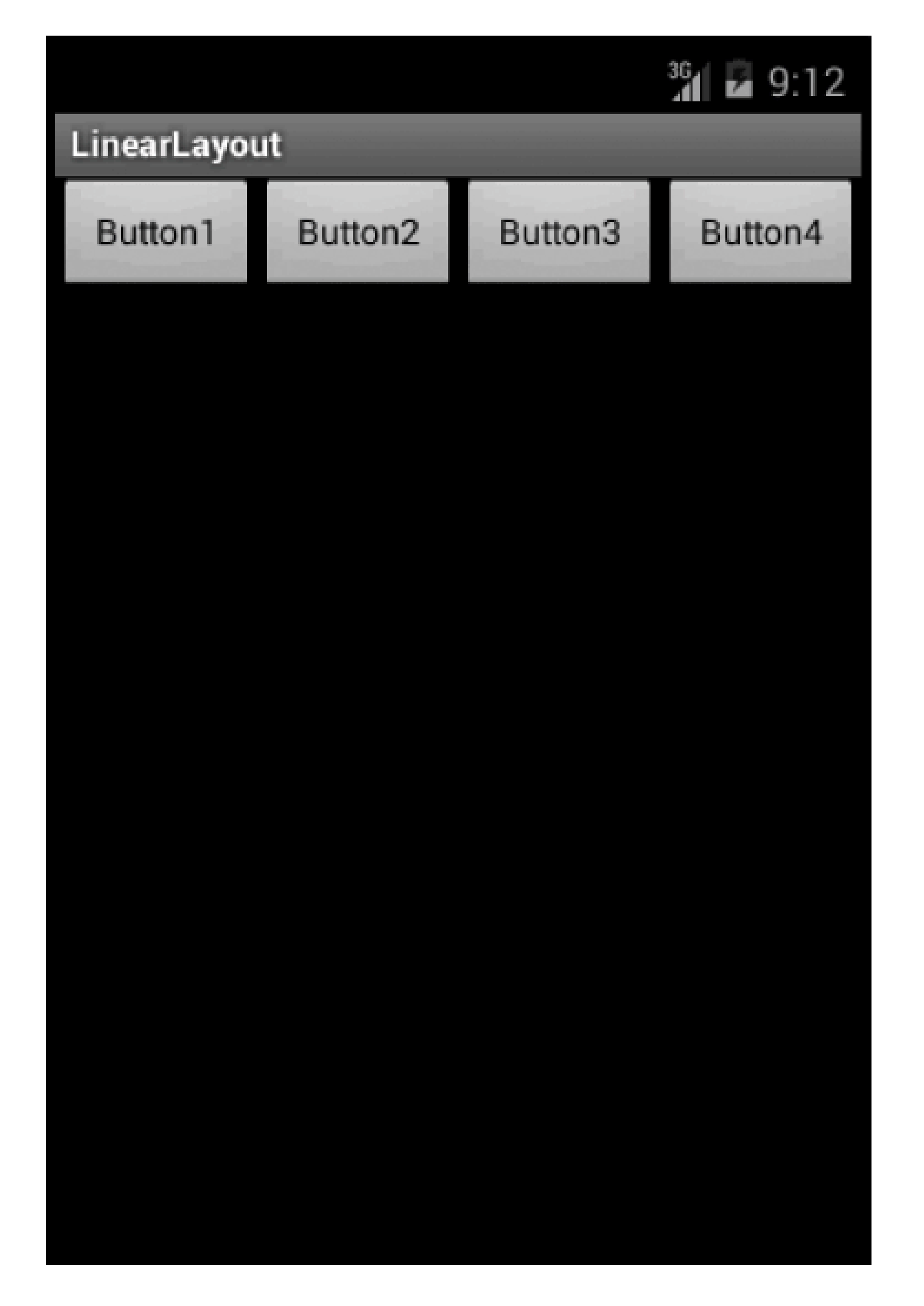


图 4.1 水平线性布局

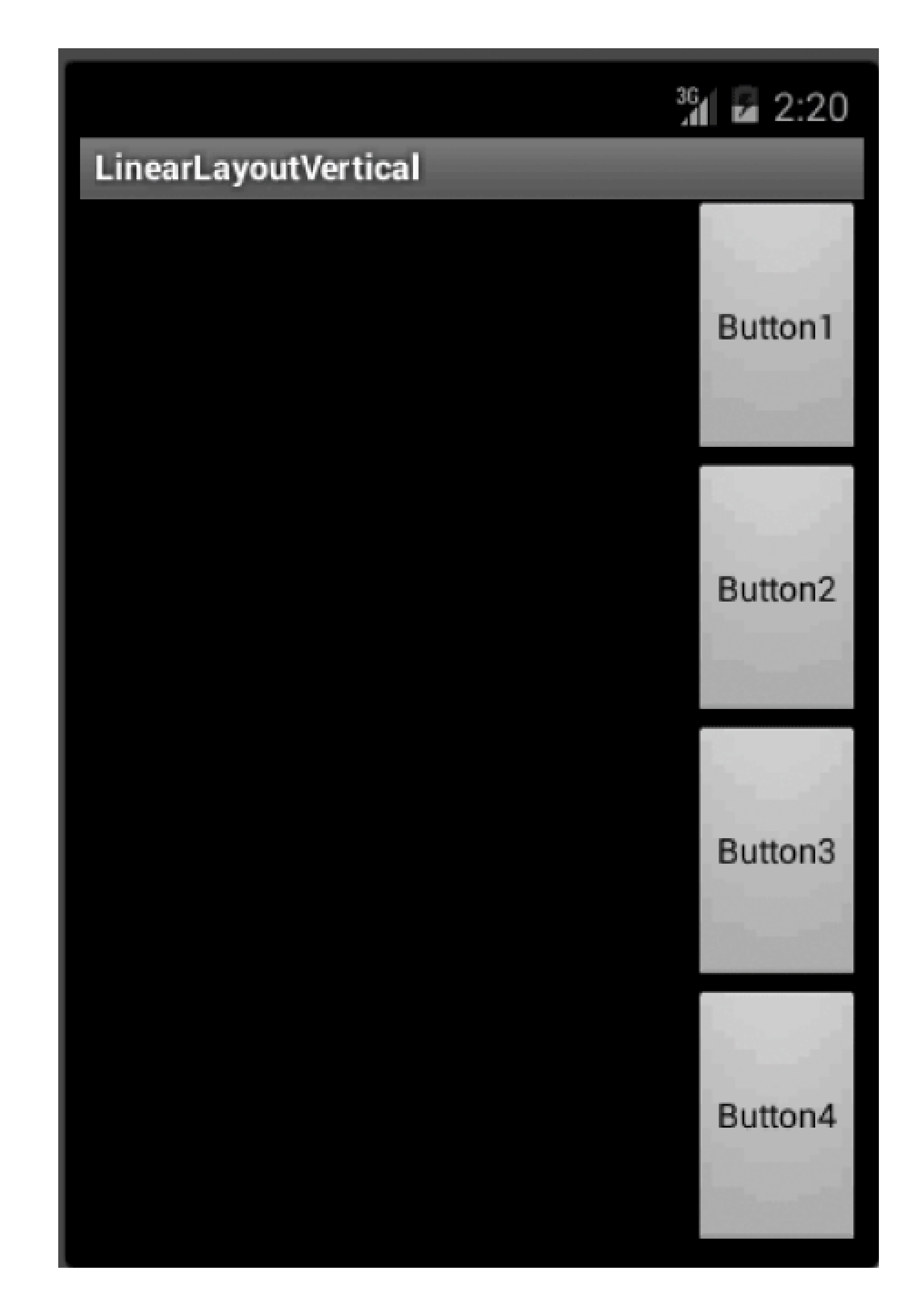


图 4.2 垂直线性布局

【例 4.2】 创建嵌套线性布局。

【解题思路】

创建嵌套线性布局，外层是垂直方向的线性布局，上下两部分高度不等；内层是水平线性布局，左右两部分宽度相等。

【开发步骤和程序分析】

(1) 在 Eclipse 中创建一个 NestLinearLayout 应用项目，包名为 com.application.nestlinearlayout。

(2) 在 res/layout 目录下的 main.xml 文件中，设置嵌套线性布局，外层是垂直方向的上下两部分高度不等的线性布局，内层是水平方向的左右两部分宽度相等的线性布局。

在该文件中编辑代码如下：

```
<?xml version = "1.0" encoding = "utf - 8"?>
<! -- 声明外层为垂直方向的线性布局 -->
< LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = "vertical"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent">
    <! -- 声明上面的内层线性布局,设置此线性布局的 layout_weight 值为 1 -->
    < LinearLayout
        android:orientation = "vertical"
```

```
10          android:layout_width = "fill_parent"
11          android:layout_height = "fill_parent"
12          android:layout_weight = "1">
13          < TextView
14              android:text = "上面的内层线性布局, layout_weight 值为 1"
15              android:layout_width = "fill_parent"
16              android:layout_height = "wrap_content"
17           />
18      </LinearLayout >
19      <! -- 声明下面的内层线性布局,设置此线性布局的 layout_weight 值为 2, -- >
20      <! -- 在此线性布局内有两个 TextView 控件,设置为水平方向  -- >
21      < LinearLayout
22          android:orientation = "horizontal"
23          android:layout_width = "fill_parent"
24          android:layout_height = "fill_parent"
25          android:layout_weight = "2">
26          android:layout_weight = "2">
27          < TextView
28              android:text = "下面的内层线性布局 1, layout_weight 值为 2"
29              android:gravity = "center_horizontal"
30              android:gravity = "center_horizontal"
31              android:background = "#aa0000"
32              android:layout_width = "wrap_content"
33              android:layout_height = "fill_parent"
34              android:layout_weight = "1"/>
35          < TextView
36              android:text = "下面的内层线性布局 2, layout_weight 值为 2"
37              android:gravity = "center_horizontal"
38              android:gravity = "center_horizontal"
39              android:background = "#00aa00"
40              android:layout_width = "wrap_content"
41              android:layout_height = "fill_parent"
42              android:layout_weight = "1"/>
43          </LinearLayout >
44  </LinearLayout >
```

① 第 2 行至第 44 行声明外层线性布局,由第 4 行得此线性布局方向是 vertical,即是垂直方向的。

② 第 8 行至第 17 行声明上面的内层线性布局,设置此线性布局的 layout_weight 值为 1,以与下面的高度不同,在此线性布局内只有一个 TextView 控件,设置为纵向。

③ 第 21 行至第 43 行声明下面的内层线性布局,设置此线性布局的 layout_weight 值为 2,其高度是上面的二分之一,在此线性布局内有两个 TextView 控件,设置为横向。

【运行结果】

在 Eclipse 中启动模拟器,然后运行项目 NestLinearLayout,嵌套线性布局运行结果如图 4.3 所示。



图 4.3 嵌套线性布局

4.2.3 表格布局

表格布局(TableLayout)通过指定行和列将界面元素添加到表格中，以多行多列的方式显示子对象，但它不会显示行、列或单元格的边框线。

表格布局的控件定义在< TableLayout ></TableLayout >标签之间，每一行为一个TableRow，每一行可以拥有0个或多个的单元格(cell)，每个单元格内是一个View对象。

表格布局常用属性如表4.3所示。

表 4.3 表格布局常用属性

属　性	说　明
Shrinkable	设置列的宽度是否可收缩，收缩指表格能够适应其父容器的大小
Stretchable	设置列的宽度是否可拉伸，拉伸指可填满表格中空余的空。
Collapsed	设置列是否被隐藏

【例 4.3】 创建表格布局。

【解题思路】

创建一个3行3列的表格布局，第1行第1列和第1列分别放1个按钮，第2行第2列和第3列分别放1个按钮，第3行第3列和第4列分别放1个按钮。

【开发步骤和程序分析】

(1) 在Eclipse中创建一个TableLayout应用项目，包名为com.application.tablelayout。

（2）在 res/layout 目录下的 main. xml 文件中设置一个 3 行 4 列的表格布局。在该文件中编辑代码如下：

```
<?xml version = "1.0" encoding = "utf - 8"?>
<! -- 声明表格布局 -->
< TableLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:shrinkColumns = "0,1,2,3">
    <! -- 声明表格的第 1 行 -->
    < TableRow ><! -- row1 -->
        < Button android:id = "@ + id/button1"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:text = "Button1"
            android:layout_column = "0"
        />
       < Button android:id = "@ + id/button2"
           android:layout_width = "wrap_content"
           android:layout_height = "wrap_content"
           android:text = "Button2"
           android:layout_column = "1"
        />
    </TableRow >
    <! -- 声明表格的第 2 行 -->
    < TableRow ><! -- row2 -->
        < Button android:id = "@ + id/button3"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:text = "Button3"
            android:layout_column = "1"
         />
        < Button android:id = "@ + id/button4"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:text = "Button4"
            android:layout_column = "2"
        />
    </TableRow >
    <! -- 声明表格的第 3 行 -->
    < TableRow ><! -- row3 -->
       < Button android:id = "@ + id/button5"
           android:layout_width = "wrap_content"
           android:layout_height = "wrap_content"
           android:text = "Button5"
```

```
43              android:layout_column = "2"
44          />
45        <Button android:id = "@ + id/button6"
46              android:layout_width = "wrap_content"
47              android:layout_height = "wrap_content"
48              android:text = "Button6"
49              android:layout_column = "3"
50          />
51      </TableRow>
52 </TableLayout>
```

① 第 2 行至第 52 行声明表格布局，第 6 行定义了表格是 4 列可收缩，布局按父容器的宽度分为 4 列，列号从 0 开始。

② 第 8 行至第 21 行声明表格的第 1 行，其中有 2 个按钮<Button>标签，第 13 行表示该行第 1 列显示按钮 1，第 19 行表示该行第 2 列显示按钮 2。

③ 第 23 行至第 36 行声明表格的第 2 行，其中有 2 个按钮<Button>标签，第 28 行表示该行第 2 列显示按钮 3，第 34 行表示该行第 3 列显示按钮 4。

④ 第 38 行至第 51 行声明表格的第 3 行，其中有 2 个按钮<Button>标签，第 43 行表示该行第 3 列显示按钮 5，第 49 行表示该行第 4 列显示按钮 6。

【运行结果】

在 Eclipse 中启动模拟器，然后运行项目 TableLayout，表格布局运行结果如图 4.4 所示。

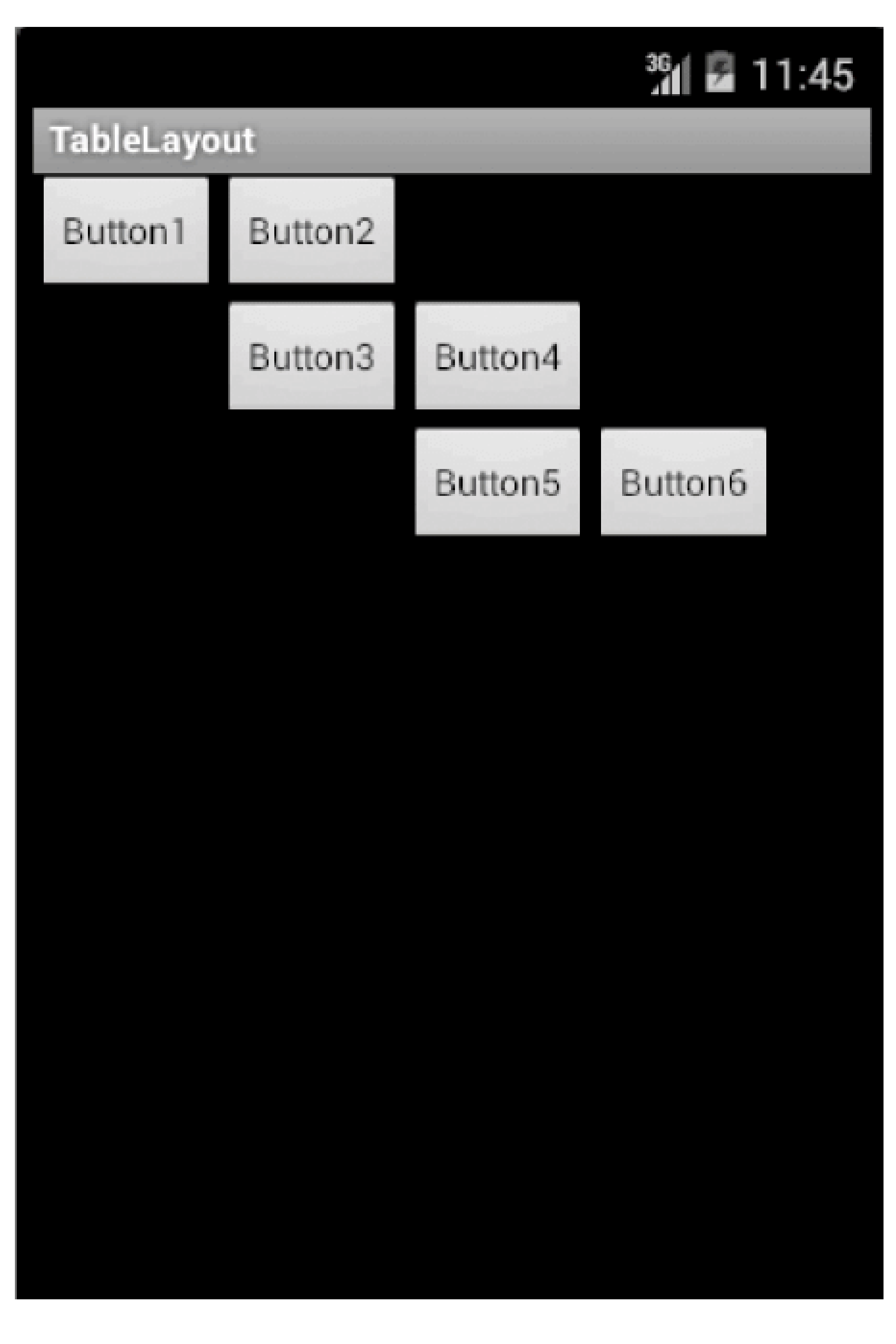


图 4.4　表格布局

4.2.4 帧布局

帧布局(FrameLayout)为加入其中的每个控件创建一个空白区域，该空白区域称为一帧，每个控件占据一帧。

帧布局的控件定义在< FrameLayout ></ FrameLayout >标签之间。

【例 4.4】 创建帧布局。

【解题思路】

创建一个帧布局，在该布局容器中添加 4 个 TextView(文本框)，这 4 个 TextView 叠加在一起，上面的 TextView 遮住下面的 TextView，通过指定 gravity 属性控制它们的对齐方式，轮换改变 4 个 TextView 背景颜色，使显示的颜色渐变不断变换，出现类似霓虹灯改变颜色的效果。

【开发步骤和程序分析】

(1) 在 Eclipse 中创建一个 FrameLayout 应用项目，包名为 com. application . framelayout。

(2) 在 res/layout 目录下的 main. xml 文件中，设置一个帧布局，使 4 个 TextView 叠加在一起。

在该文件中编辑代码如下：

```
<?xml version = "1.0" encoding = "utf - 8"?>
    <! -- 声明帧布局 -->
    < FrameLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    >
    <! -- 声明第 1 个 TextView,位于底层 -->
    < TextView android:id = "@ + id/view01"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:layout_gravity = "center"
        android:width = "320px"
        android:height = "320px"
        android:background = " # f00"
    />
    <! -- 声明第 2 个 TextView,位于第 1 个 TextView 的上层 -->
    < TextView android:id = "@ + id/view02"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:layout_gravity = "center"
        android:width = "280px"
        android:height = "280px"
        android:background = " # 0f0"
    />
```

```
25     <!-- 声明第 3 个 TextView,位于第 2 个 TextView 的上层 -->
26     <TextView android:id = "@ + id/view03"
27         android:layout_width = "wrap_content"
28           android:layout_height = "wrap_content"
29           android:layout_gravity = "center"
30           android:width = "240px"
31           android:height = "240px"
32           android:background = "#00f"
33     />
34     <!-- 声明第 4 个 TextView,位于顶层 -->
35     <TextView android:id = "@ + id/view04"
36           android:layout_width = "wrap_content"
37           android:layout_height = "wrap_content"
38           android:layout_gravity = "center"
39           android:width = "200px"
40           android:height = "200px"
41         android:background = "#ff0"
42    />
43 </FrameLayout>
```

① 第 2 行至第 43 行声明帧布局。

② 第 8 行至第 42 行声明 4 个 TextView,先定义的 TextView 位于底层,后定义的 TextView 位于上层。

③ 第 8 行至第 15 行声明第 1 个 TextView 位于底层,第 11 行 layout_gravity 值为 center,表示对齐方式为居中,第 12 行表示宽度为 320px,第 13 行表示高度为 320px,第 14 行 background 值为#f00,表示背景颜色为红色。

(3) 在 src/com.application.framelayout 包下的 FrameLayoutActivity.java 文件中,加载 main.xml 布局文件,启动一条线程来周期性地改变 4 个 TextView 的背景颜色。

在该文件中编辑代码如下:

```
package com.application.framelayout;

import java.util.Timer;
import java.util.TimerTask;
import com.application.framelayout.R;
import android.app.Activity;
import android.os.Bundle;
import android.os.Handler;
import android.os.Message;
import android.widget.TextView;

public class FrameLayoutActivity extends Activity {
    private int currentColor = 0;
    //定义 1 个颜色数组
```

```
    final int[] colors = new int[] {
        R.color.color1,
        R.color.color2,
        R.color.color3,
        R.color.color4
    };

    inal int[] names = new int[] {
        R.id.view01,
        R.id.view02,
        R.id.view03,
        R.id.view04
   };
    TextView[] views = new TextView[names.length];
    //定义 1 个 Handler 来处理 TextView 的背景颜色的更新
    Handler handler = new Handler() {

        @Override
        public void handleMessage(Message msg) {
            if (msg.what == 0x123) {
               for (int i = 0; i < names.length; i++) {
                     views[i].setBackgroundResource(colors[(i
                             + currentColor)  % names.length]);
                }
            currentColor++;
            }
            super.handleMessage(msg);
        }
   };
   //定义一个每 0.2 秒执行一次的任务,该任务向 Handler 发送一条消息,
   //通知它更新 4 个 TextView 的背景颜色
   @Override
   public void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.main);
            for (int i = 0; i < names.length; i++) {
                views[i] = (TextView) findViewById(names[i]);
            }
            new Timer().schedule(new TimerTask() {

                    @Override
                    public void run() {
                        handler.sendEmptyMessage(0x123);
                    }
                }, 0, 200);
```

```
60      }
61 }
```

① 第 1 行至第 61 行，该 Java 程序使用上述帧布局，启动一条线程来周期性地改变 4 个 TextView 的背景颜色。

② 第 15 行至第 20 行定义一个颜色数组。

③ 第 30 行至第 43 行定义一个 Handler 来处理 TextView 的背景颜色的更新。

④ 第 45 行至第 59 行定义一个每 0.2 秒执行一次的任务，该任务向 Handler 发送一条消息，通知它更新 4 个 TextView 的背景颜色。

【运行结果】

在 Eclipse 中启动模拟器，然后运行项目 FrameLayout，帧布局运行结果如图 4.5 所示。

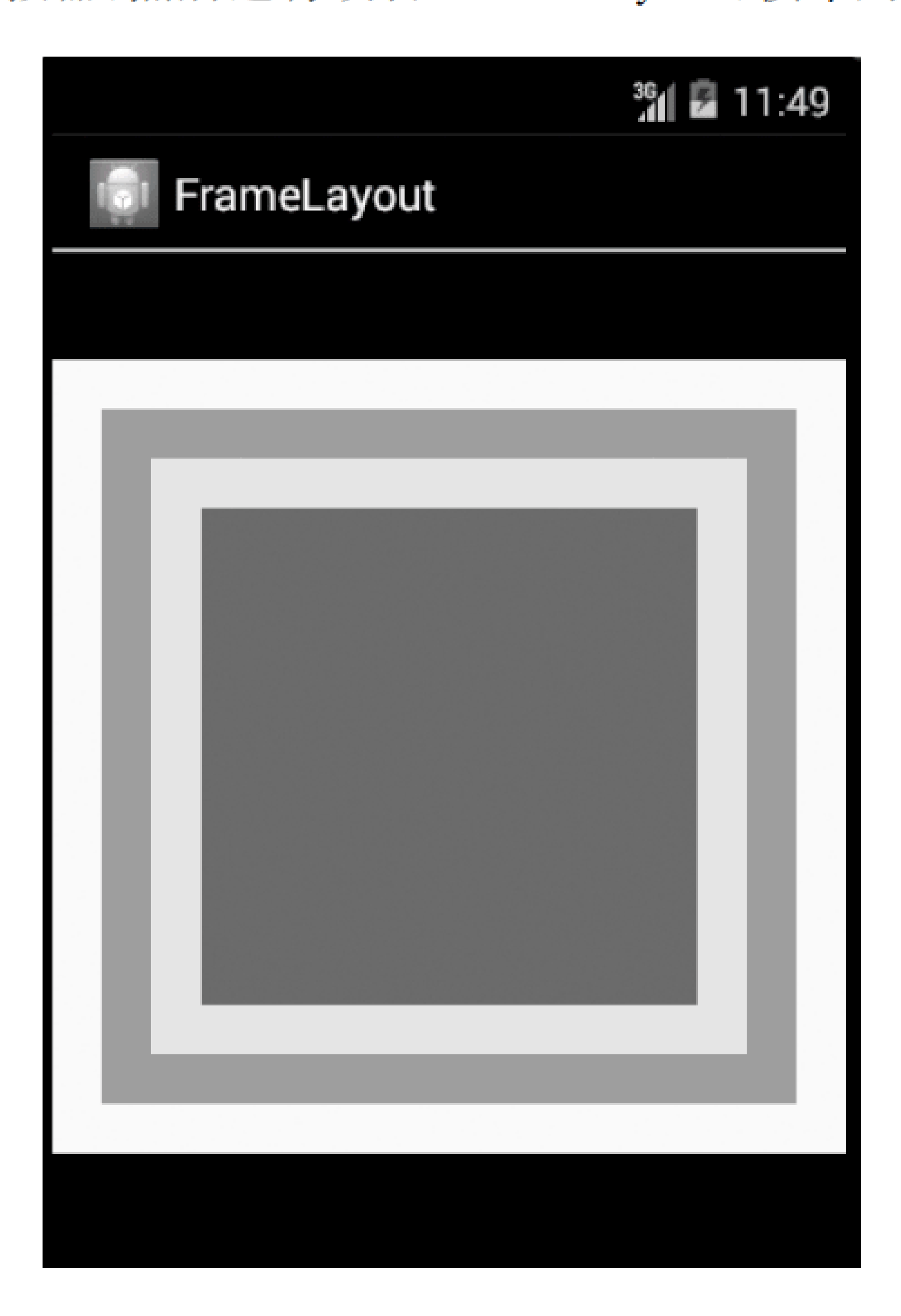


图 4.5　帧布局

4.2.5　网格布局

网格布局(GridLayout)是 Android SDK 4.0 新增加的布局方式，它将用户界面划分为网格，界面元素可随意摆放在网格中，网格布局比表格布局在界面设计上更加灵活，在网格布局中界面元素可以占用多个网格。

网格布局的控件定义在< GridLayout ></GridLayout >标签之间。

【例 4.5】　创建网格布局。

【解题思路】

创建网格布局，实现计算器界面，将界面划分为 6×4 的网格。其中，第 1 个文本框横跨

4 列,以下每一个按钮各占一格。

【开发步骤和程序分析】

(1) 在 Eclipse 中创建一个 GridLayout 应用项目,包名为 com. application. gridlayout。

(2) 在 res/layout 目录下的 main. xml 文件中,设置一个网格布局,将界面划分为 6 行 4 列的网格以实现计算器界面。

在该文件中编辑代码如下:

```
1  <?xml version = "1.0" encoding = "utf - 8" ?>
2  <! -- 声明网格布局 -- >
3  < GridLayout xmlns:android = "http://schemas. android. com/apk/res/android"
4      android:layout_width = "match_parent"
5      android:layout_height = "match_parent"
6      android:rowCount = "6"
7      android:columnCount = "4"
8      android:id = "@ + id/root"
9      >
10      <! -- 定义一个横跨 4 列的文本框 -- >
11     < TextView
12         android:layout_width = "match_parent"
13         android:layout_height = "wrap_content"
14         android:layout_columnSpan = "4"
15         android:textSize = "50sp"
16         android:layout_marginLeft = "4px"
17         android:layout_marginRight = "4px"
18         android:padding = "5px"
19         android:layout_gravity = "right"
20         android:background = "#eee"
21         android:textColor = "#000"
22         android:text = "0"/>
23 </GridLayout >
```

① 第 2 行至第 23 行声明网格布局。

② 第 6 行设置行数为 6,第 7 行设置列数为 4。

③ 第 11 行至第 22 行定义一个横跨 4 列的文本框,并设置该文本框的前景色、背景色等属性。

(3) 在 src/com. application. gridlayout 包下的 GridLayoutActivity. java 文件中,加载 main. xml 布局文件,使用程序循环向网格布局添加 20 个按钮。

在该文件中编辑代码如下:

```
package com. application. gridlayout;

import com. application. gridlayout. R;
import android. os. Bundle;
import android. app. Activity;
```

```
6  import android.view.Gravity;
7  import android.widget.Button;
8  import android.widget.GridLayout;
9  //该 Java 程序使用循环向 GridLayout 添加 20 个按钮
10 public class GridLayoutActivity extends Activity{
11     GridLayout gridLayout;
12     //用数组定义 20 个按钮的文本
13     String[ ] chars = new String[ ] {
14             "C" , "( )" , " + / - " , "Del",
15             "7" , "8" , "9" , " ÷ ",
16             "4" , "5" , "6" , " x ",
17             "1" , "2" , "3" , " - ",
18             "0" , "." , " = " , " + "
19     };
20
21    @Override
22    public void onCreate(Bundle savedInstanceState) {
23         super.onCreate(savedInstanceState);
24         setContentView(R.layout.main);
25         gridLayout = (GridLayout) findViewById(R.id.root);
26         //使用循环向 GridLayout 添加 20 个按钮
27         for(int i = 0 ; i < chars.length ; i++) {
28                 Button bn = new Button(this);
29                 bn.setText(chars[i]);
30                 bn.setTextSize(40);
31                 GridLayout.Spec rowSpec = GridLayout.spec(i / 4 + 1);
32                 GridLayout.Spec columnSpec = GridLayout.spec(i % 4);
33                 GridLayout.LayoutParams params = new GridLayout.LayoutParams(
34                     rowSpec , columnSpec);
35                 params.setGravity(Gravity.FILL);
36                 gridLayout.addView(bn , params);
37         }
38     }
39 }
```

① 第 10 行至第 39 行，该 Java 程序使用循环向 GridLayout 添加 20 个按钮。

② 第 13 行至第 19 行用数组定义 20 个按钮的文本。

③ 第 26 行至第 37 行使用循环向 GridLayout 添加 20 个按钮，第 30 行设置该按钮的字体大小，第 31 行指定该组件所在的行，第 32 行指定该组件所在的列。

【运行结果】

在 Eclipse 中启动模拟器，然后运行项目 GridLayout，网格布局运行结果如图 4.6 所示。

4.2.6 相对布局

相对布局(RelativeLayout)是一种灵活的布局方式，能够通过指定界面元素与其他元素

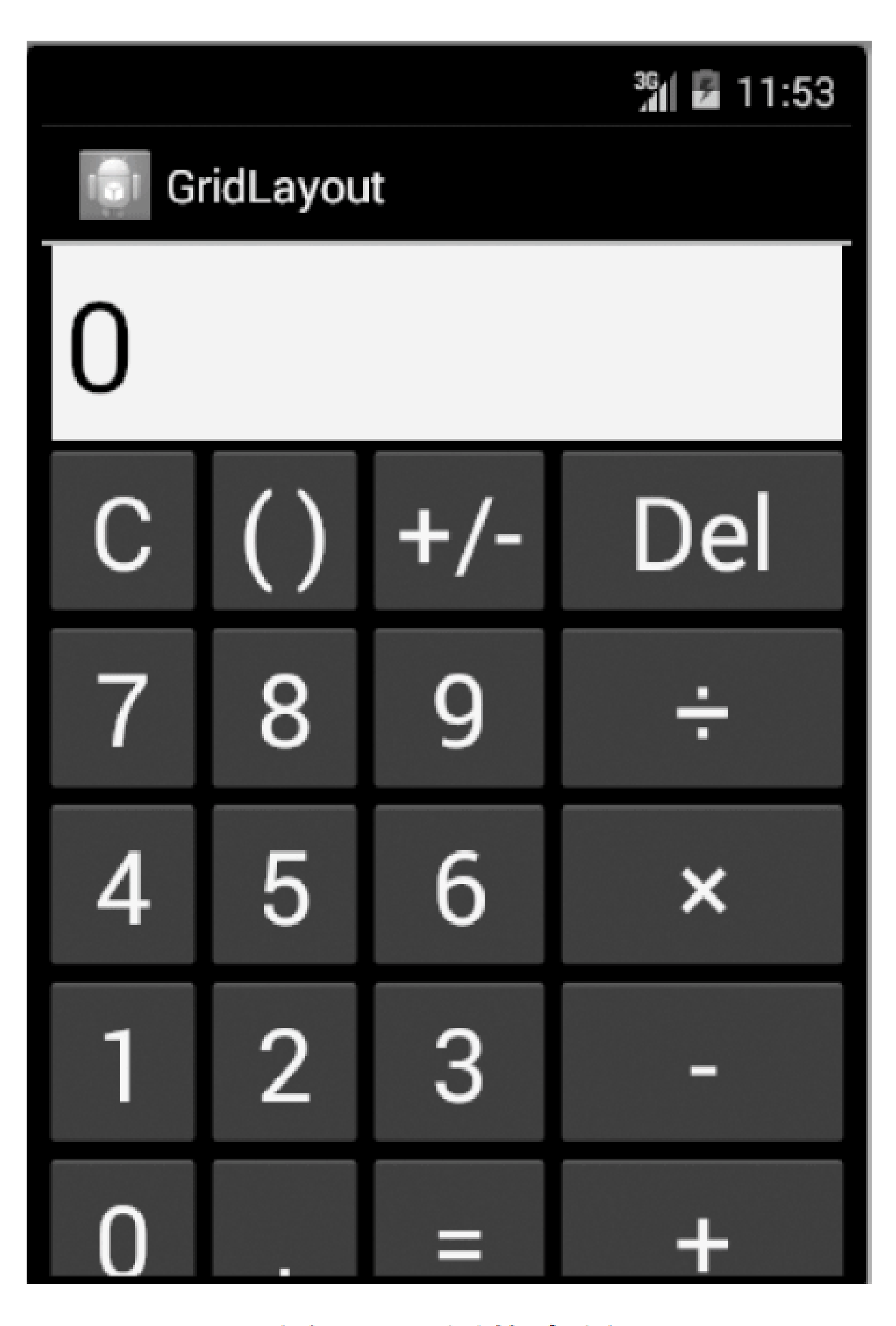


图 4.6 网格布局

的相对位置关系，确定界面中所有元素的布局位置，从而保证在各种屏幕尺寸的手机上正确显示界面布局。

相对布局的控件定义在<RelativeLayout></RelativeLayout>标签之间。

【例 4.6】 创建相对布局。

【解题思路】

创建相对布局，上方是一个与屏幕左对齐的字符串，下面是一个可输入的编辑框，在它的下方是两个与屏幕右对齐的按钮。

【开发步骤和程序分析】

(1) 在 Eclipse 中创建一个 RelativeLayout 应用项目，包名为 com.application.relativelayout。

(2) 在 res/layout 目录下的 main.xml 文件中，设置相对布局，最上方是一个与屏幕左对齐的文本框，文本框下面是一个可输入的编辑框，编辑框的下面是与屏幕右对齐的 Reset 按钮，Reset 按钮的左边是 OK 按钮。

在该文件中编辑代码如下：

```
<?xml version = "1.0" encoding = "utf - 8"?>
<! -- 声明相对布局 -->
<RelativeLayout xmlns:android = "http://schemas.android.com/apk/res/android"
          android:layout_width = "fill_parent"
          android:layout_height = "fill_parent">
    <! -- 声明一个 TextView 控件,其 ID 为"label" -->
```

```
7       <TextView
8           android:id="@+id/label"
9           android:layout_width="fill_parent"
10           android:layout_height="wrap_content"
11          android:text="相对布局"
12       />
13      <!-- 声明一个 EditText 控件,其 ID 为 entry,设置本编辑框位置为 -->
14      <!-- ID 为"label"的 TextView 控件的下方 -->
15     <EditText
16          android:id="@+id/entry"
17          android:layout_width="fill_parent"
18          android:layout_height="wrap_content"
19          android:background="@android:drawable/editbox_background"
20          android:layout_below="@id/label"/>
21      <!-- 声明一个 Button 控件,其 ID 为 reset,设置本按钮位置为 -->
22      <!-- ID 为 entry 的 EditText 控件的下方 -->
23      <Button
24          android:id="@+id/reset"
25          android:layout_width="wrap_content"
26          android:layout_height="wrap_content"
27          android:layout_below="@id/entry"
28          android:layout_alignParentRight="true"
29          android:layout_marginLeft="10dip"
30         android:text="Reset" />
31     <!--行声明一个 Button 控件,设置本按钮位置为 ID 为 reset 的 Button 控件的左侧-->
32     <Button
33          android:layout_width="wrap_content"
34          android:layout_height="wrap_content"
35          android:layout_toLeftOf="@id/ reset "
36          android:layout_alignTop="@id/ reset "
37          android:text="   OK " />
38 </RelativeLayout>
```

① 第 2 行至第 38 行声明相对布局。

② 第 7 行至第 12 行声明一个<TextView>控件,其 ID 为“label”,第 9 行设置文本框宽度为 fill_parent,即填满其父容器,第 10 行设置文本框高度为 wrap_content,即其高度为文本高度。

③ 第 15 行至第 20 行声明一个<EditText>控件,其 ID 为“entry”,第 20 行设置本编辑框位置为 ID 为“abel”的<TextView>控件的下方。

④ 第 23 行至第 30 行声明一个<Button>控件,其 ID 为“reset”,第 27 行设置本按钮位置为 ID 为“entry”的<EditText>控件的下方,第 28 行指定它与父容器的右侧对齐,第 29 行指定本按钮左侧留白为 10dip。

⑤ 第 32 行至第 37 行声明一个<Button>控件,第 35 行设置本按钮位置为 ID 为“reset”的按钮控件的左侧,第 36 行指定它与 Reset 按钮控件的上边界对齐。

【运行结果】

在 Eclipse 中启动模拟器,然后运行项目 RelativeLayout,相对布局运行结果如图 4.7 所示。

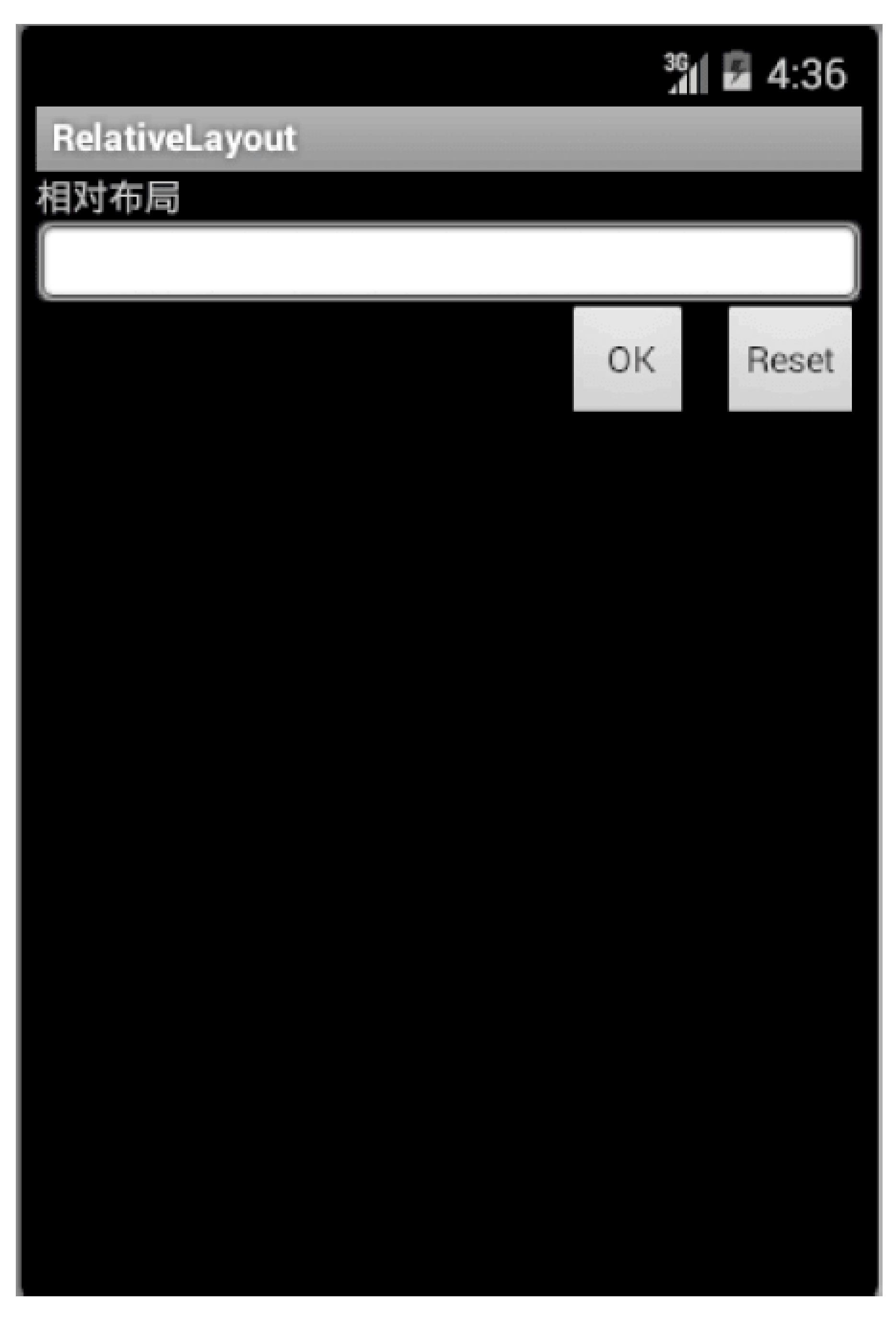


图 4.7　相对布局

4.2.7　绝对布局

绝对布局(AbsoluteLayout)中所有控件的位置通过设置控件的坐标(x,y)来指定,坐标原点(0,0)在屏幕左上角。

绝对布局的控件定义在< AbsoluteLayout ></AbsoluteLayout >标签之间。

通过坐标确定元素位置后,系统无法根据不同屏幕大小对元素位置进行调整,降低了布局对不同类型和尺寸屏幕的适应能力,因此,不推荐使用绝对布局方式。

【例 4.7】 创建绝对布局。

【解题思路】

创建绝对布局,屏幕上方是一个居中的字符串,下面是一个可输入的编辑框,在它的下方是两个与屏幕左对齐的按钮。

【开发步骤和程序分析】

(1) 在 Eclipse 中创建一个 AbsoluteLayout 应用项目,包名为 com. application. absolutelayout。

(2) 在 res/layout 目录下的 main. xml 文件中,设置绝对布局,最上方是一个与屏幕左对齐的文本框,文本框下面是一个编辑框,编辑框的下面是与屏幕左对齐的 OK 按钮和

Reset 按钮，它们的位置都通过控件的坐标指定。

在该文件中编辑代码如下：

```
1  <?xml version = "1.0" encoding = "utf - 8"?>
2  <! -- 声明绝对布局 -->
3  < AbsoluteLayout android:id = "@ + id/absolutelayout01"
4      android:layout_width = "fill_parent"
5      android:layout_height = "fill_parent"
6      xmlns:android = "http://schemas.android.com/apk/res/android">
7      <! -- 声明一个 TextView 控件 -->
8      < TextView android:id = "@ + id/label"
9              android:layout_x = "130dip"
10             android:layout_y = "10dip"
11             android:layout_height = "wrap_content"
12             android:layout_width = "wrap_content"
13             android:text = "绝对布局">
14     </TextView >
15     <! -- 声明一个 EditText 控件 -->
16     < EditText android:id = "@ + id/entry"
17             android:layout_x = "10dip"
18             android:layout_y = "30dip"
19             android:layout_height = "wrap_content"
20             android:layout_width = "300dip"
21             android:background = "@android:drawable/editbox_background" >
22     </EditText >
23     <! -- 声明一个 Button 控件 -->
24     < Button android:id = "@ + id/ok"
25             android:layout_width = "60dip"
26             android:layout_height = "wrap_content"
27             android:layout_x = "10dip"
28             android:layout_y = "60dip"
29             android:text = "OK">
30     </Button >
31     <! -- 声明一个 Button 控件 -->
32     < Button android:id = "@ + id/reset "
33             android:layout_width = "70dip"
34             android:layout_height = "wrap_content"
35             android:layout_x = "80dip"
36             android:layout_y = "60dip"
37             android:text = "Reset" >
38     </Button >
39 </AbsoluteLayout >
```

① 第 2 行至第 39 行声明绝对布局。

② 第 8 行至第 14 行声明一个< TextView >控件，第 9 行设置文本框 X 坐标为 130dip，

第 10 行设置文本框 Y 坐标为 10dip。

③ 第 16 行至第 22 行声明一个<EditText>控件，第 17 行设置编辑框 X 坐标为 10dip，第 18 行设置编辑框 Y 坐标为 30dip。

④ 第 24 行至第 30 行声明一个<Button>控件，第 27 行设置本按钮 X 坐标为 10dip，第 28 行设置本按钮 Y 坐标为 60dip。

⑤ 第 32 行至第 38 行声明一个<Button>控件，第 35 行设置本按钮 X 坐标为 80dip，第 36 行设置本按钮 Y 坐标为 60dip。

【运行结果】

在 Eclipse 中启动模拟器，然后运行项目 AbsoluteLayout，水平线性布局运行结果如图 4.8 所示。



图 4.8　绝对布局

4.3　常用控件

Widget(部件)是为构建用户交互界面而提供服务的视图对象，Widget 类是 View 类的子类。

Android 的可视控件都在 android.widget 包内。Widget 常用的控件包括：TextView、EditText、Button、ImageButton、Checkbox、RadioButton、ImageView、AnalogClock、DigitalClock、DatePicker、TimePicker 等，下面分别介绍。

4.3.1 TextView

TextView(文本框)用于向用户显示文本内容,其显示的文本只能在初始设置时或在程序中修改。

TextView 继承自 View,TextView 在 android. widget. TextView 包中定义,在 Java 程序设计中使用时,在相应代码文件前部引入该包,语句为“import android. widget. TextView;”。

TextView 常用属性如表 4.4 所示。

表 4.4 TextView 常用属性

属　　性	说　　明
gravity	定义横向和纵向的显示方式
height	以像素为单位定义高度
width	以像素为单位定义宽度
text	显示的文本内容
textSize	设置显示的文本大小
textColor	设置显示的文本颜色
typeface	设置显示的文本字体

尺寸单位如表 4.5 所示。

表 4.5 尺寸单位

尺寸单位名称	说　　明
px(像素)	屏幕上的像素
dip(又名 dp,与密度无关的像素)	在不同密度的屏幕中显示比例保持一致,密度是屏幕每英寸所包含的像素数
sp(可伸缩像素)	用于文字大小的适配问题
in (英寸)	长度单位,1in=25.4mm
pt (磅)	1/72 英寸
mm(毫米)	长度单位

Android 颜色值由红(Red)、绿(Green)、蓝(Blue)三原色和一个透明度(Alpha)值来表示,颜色值以“#”开头,接着是 Alpha-Red-Green-Blue,如果省略 Alpha 值,该颜色默认是完全不透明。

Android 颜色值形式有:#RGB(三位十六进制数)、#RRGGBB(六位十六进制数)、#ARGB、#AARRGGBB 等。例如,Red(红色)的 #RGB 值为 #F00,#RRGGBB 值为 #FF0000。

常用颜色值如表 4.6 所示。

表 4.6 常用颜色值

颜 色 名 称	#RRGGBB	颜 色 名 称	#RRGGBB
Black(黑色)	#000000	Purple(紫色)	#800080
Navy(深蓝色)	#000800	Olive(橄榄绿)	#808000
Blue(蓝色)	#0000FF	Gray(灰色)	#808080
Green(绿色)	#008000	Silver(银色)	#C0C0C0
Teal(青色)	#008080	Red(红色)	#FF0000
Lime(绿黄色)	#00FF00	Fuchsia(紫红色)	#FF00FF
Aqua(浅绿色)	#00FFFF	Yellow(黄色)	#FFFF00
Maroon(酱紫色)	#800000	White(白色)	#FFFFFF

【例 4.8】 TextView 控件举例。

【解题思路】

创建 TextView 控件，在 5 个 TextView 中，分别设置字号、文本大小、颜色、图片和链接等。

【开发步骤和程序分析】

(1) 在 Eclipse 中创建一个 TextViewExample 应用项目，包名为 com.application.textviewexample。

(2) 在 res/layout 目录下的 main.xml 文件中，设置 5 个 TextView 控件，分别设置字号、文本大小、颜色、图片和链接等。

在该文件中编辑代码如下：

```
<?xml version = "1.0" encoding = "utf - 8"?>
<! -- 声明一个垂直线性布局 -->
< LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
          android:orientation = "vertical"
          android:layout_width = "fill_parent"
          android:layout_height = "fill_parent"
          android:background = "@drawable/background">
          <! -- 声明第 1 个 TextView 控件,设置字号为 6pt -->
          < TextView
              android:layout_width = "fill_parent"
              android:layout_height = "wrap_content"
              android:text = "文本框示例: 显示不同字号、颜色和带链接的文本"
              android:textSize = "6pt"/>
          <! -- 声明第 2 个 TextView 控件,设置文本颜色为绿色 -->
          < TextView
              android:id = "@ + id/textView2"
              android:textColor = " #0f0"
              android:textSize = "20px"
              android:text = "测试文本: 20px, 绿色"
              android:width = "300px"
              android:layout_width = "wrap_content"
              android:layout_height = "wrap_content" />
          <! -- 声明第 3 个 TextView 控件,设置文本颜色为红色 -->
```

```
24          <TextView
25              android:id = "@ + id/textView3"
26              android:textColor = " # f00"
27              android:textSize = "30px"
28              android:text = "测试文本: 30px, 红色"
29              android:width = "300px"
30              android:singleLine = "true"
31              android:layout_width = "wrap_content"
32              android:layout_height = "wrap_content" />
33          <! -- 声明第 4 个 TextView 控件,设置文字大小为 10pt,绘制图片 -->
34          <TextView
35              android:layout_width = "fill_parent"
36              android:layout_height = "wrap_content"
37              android:text = "Knowledge is power !"
38              android:textSize = "10pt"
39              android:drawableEnd = "@drawable/ic_launcher"/>
40          <! -- 声明第 5 个 TextView 控件,显示邮件、电话,并对邮件添加链接 -->
41          <TextView
42              android:layout_width = "fill_parent"
43              android:layout_height = "wrap_content"
44              android:singleLine = "true"
45              android:text = "邮件是 liming@uestc.edu.cn,电话是 02888888888"
46              android:autoLink = "email|phone"/>
47  </LinearLayout>
```

① 第 2 行至第 47 行声明一个垂直线性布局,其中有 5 个 TextView 控件。

② 第 9 行至第 13 行声明第 1 个 TextView 控件,第 13 行设置字号为 6pt。

③ 第 15 行至第 22 行声明第 2 个 TextView 控件,第 17 行设置文本颜色为绿色,第 18 行设置文本大小为 20px。

④ 第 24 行至第 32 行声明第 3 个 TextView 控件,第 26 行设置文本颜色为红色,第 27 行设置文本大小为 30px。

⑤ 第 24 行至第 39 行声明第 4 个 TextView 控件,第 38 行设置文字大小为 10pt,第 39 行绘制图片。

⑥ 第 41 行至第 46 行声明第 5 个 TextView 控件,第 45 行显示邮件、电话,并对邮件添加链接。

【运行结果】

在 Eclipse 中启动模拟器,然后运行项目 TextViewExample,水平线性布局运行结果如图 4.9 所示。



图 4.9　TextView 控件举例

4.3.2 EditText

EditText(编辑框)用于显示文本的内容,并可对文本进行编辑,而 TextView 仅用于设置显示的文本。EditText 是一个具有编辑功能的 TextView,EditText 是 TextView 的子类。

EditText 定义在 android.widget.EditText 包中,在 Java 程序设计中使用时,在相应代码文件前部引入该包,语句为“import android.widget.EditText;”。

EditText 常用属性如表 4.7 所示。

表 4.7 EditText 常用属性

属　性	说　明
cursorVisible	设置光标是否可见,默认可见
lines	设置固定行数以确定 EditText 的高度
maxLines	设置最大的行数
singleLine	设置文本框为单行模式
maxLength	设置最大的显示长度
scrollHorizontally	设置文本框是否可以进行水平滚动
password	设置显示是否为密码模式
phoneNumber	设置显示文本只能是电话号码

【例 4.9】 EditText 控件举例。

【解题思路】

创建 EditText 控件,在 4 个 EditText 中,分别设置登录账号、输入密码、确认密码、E-mail。

【开发步骤和程序分析】

(1) 在 Eclipse 中创建一个 EditTextExample 应用项目,包名为 com.application.edittextexample。

(2) 在 res/layout 目录下的 main.xml 文件中,设置 4 个 EditText,分别设置登录账号、输入密码、确认密码、E-mail。

在该文件中编辑代码如下:

```
<?xml version = "1.0" encoding = "utf - 8"?>
<! -- 声明一个垂直线性布局 -->
< TableLayout xmlns:android = "http://schemas.android.com/apk/res/android"
            android:id = "@ + id/tableLayout1"
            android:layout_width = "fill_parent"
            android:layout_height = "fill_parent"
            android:background = "@drawable/background">
            < TableRow android:id = "@ + id/tableRow1"
                android:layout_width = "wrap_content"
              android:layout_height = "wrap_content">
```

```
<TextView
    android:layout_width = "wrap_content"
    android:layout_height = "wrap_content"
    android:text = "用户名: "
    android:height = "50px" />
<!-- 声明第 1 个 EditText 控件,用于输入登录账号 -->
<EditText android:id = "@ + id/nickname"
    android:hint = "请填写登录账号"
    android:layout_width = "300px"
    android:layout_height = "wrap_content"
    android:singleLine = "true"
/>
</TableRow>
<TableRow android:id = "@ + id/tableRow2"
    android:layout_width = "wrap_content"
    android:layout_height = "wrap_content">
<TextView
    android:layout_width = "wrap_content"
    android:layout_height = "wrap_content"
    android:text = "输入密码: "
    android:height = "50px" />
<!-- 声明第 2 个 EditText 控件,用于输入密码 -->
<EditText android:id = "@ + id/pwd"
    android:layout_width = "300px"
    android:inputType = "textPassword"
    android:layout_height = "wrap_content"
    />
</TableRow>
<TableRow android:id = "@ + id/tableRow3"
    android:layout_width = "wrap_content"
    android:layout_height = "wrap_content">
< TextView
    android:layout_width = "wrap_content"
    android:layout_height = "wrap_content"
    android:text = "确认密码: "
    android:height = "50px" />
<!-- 声明第 3 个 EditText 控件,用于确认密码 -->
<EditText android:id = "@ + id/repwd"
    android:layout_width = "300px"
    android:layout height = "wrap_content"
    android:inputType = "textPassword"
    />
</TableRow>
<TableRow android:id = "@ + id/tableRow4"
    android:layout_width = "wrap_content"
```

```
56                android:layout_height = "wrap_content">
57            < TextView
58               android:layout_width = "wrap_content"
59                android:layout_height = "wrap_content"
60                android:text = "E - mail: "
61                android:height = "50px" />
62            <! -- 声明第 4 个 EditText 控件,用于输入 E - mail -->
63            < EditText android:id = "@ + id/email"
64                android:layout_width = "300px"
65                android:layout_height = "wrap_content"
66                android:inputType = "textEmailAddress"
67                />
68            </TableRow >
69            < LinearLayout
70                android:orientation = "horizontal"
71                android:layout_width = "wrap_content"
72                android:layout_height = "wrap_content" >
73                < Button android:text = "注册"
74                   android:id = "@ + id/button1"
75                   android:layout_width = "wrap_content"
76                   android:layout_height = "wrap_content"/>
77                < Button android:text = "重置"
78                   android:id = "@ + id/button2"
79                   android:layout_width = "wrap_content"
80                   android:layout_height = "wrap_content"/>
81            </LinearLayout >
82 </TableLayout >
```

① 第 2 行至第 82 行声明一个表格布局,其中有 4 个 TextView 控件,4 个 EditText 控件,并嵌套了 1 个水平线性布局,在该线性布局中,有 2 个 Button 控件。

② 第 17 行至第 22 行声明第 1 个 EditText 控件,用于输入登录账号。

③ 第 33 行至第 37 行声明第 2 个 EditText 控件,用于输入密码,第 35 行设置密码框,只能接受文本密码输入。

④ 第 48 行至第 52 行声明第 3 个 EditText 控件,用于确认密码,第 51 行设置密码框,只能接受文本密码输入。

⑤ 第 63 行至第 67 行声明第 4 个 EditText 控件,用于输入 E-mail,第 66 行设置电子邮件框,只能接受电子邮件输入。

【运行结果】

在 Eclipse 中启动模拟器,然后运行项目 EditTextExample,EditText 控件举例运行结果如图 4.10 所示。

4.3.3 Button 和 ImageButton

Button(按钮)是广泛使用的控件,它继承自 TextView。Button 上可显示文本,它的背



图 4.10　EditText 控件举例

景可以改变颜色，或显示图片。

当用户按下按钮或单击后，就会触发 onClick 事件，需要对 Button 控件设置 OnClickListener 监听器，在按钮监听器的实现代码中编写按钮按下事件的处理代码。

Button 定义在 android.widget.Button 包中，在 Java 程序设计中使用时，在相应代码文件前部引入该包，语句为“import android.widget.Button;”。

ImageButton（图片按钮）控件和 Button 控件功能一致，但显示效果不同，在 ImageButton 的按钮中只显示图片，不显示文本。

ImageButton 继承自 ImageView，为了不影响图片的显示，一般要将其背景色设置为透明，ImageButton 在未被按下和被按下不同状态通常要设置不同的图片，以示区别。

【例 4.10】　Button 和 ImageButton 控件举例。

【解题思路】

创建 Button 和 ImageButton 控件，有一个按钮和一个图片按钮。

【开发步骤和程序分析】

(1) 在 Eclipse 中创建一个 ButtonExample 应用项目，包名为 com.application.buttonexample。

(2) 在 res/layout 目录下的 main.xml 文件中，设置 Button 和 ImageButton 控件，即一个按钮和一个图片按钮。

在该文件中编辑代码：

```
<?xml version = "1.0" encoding = "utf - 8"?>
```

```
<!-- 声明一个垂直线性布局 -->
<LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
     android:orientation = "vertical"
     android:layout_width = "fill_parent"
     android:layout_height = "fill_parent">
        <TextView android:id = "@ + id/TextView01"
            android:layout_width = "fill_parent"
            android:layout_height = "wrap_content"
             android:text = "@string/hello"/>
         <!-- 声明一个按钮 -->
         <Button android:id = "@ + id/Button01"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:text = "Button01" >
        </Button>
        <!-- 声明一个图片按钮 -->
        <ImageButton android:id = "@ + id/ImageButton01"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:text = "ImageButton01">
        </ImageButton>
</LinearLayout>
```



图 4.11　Button 和 ImageButton 控件

① 第 2 行至第 20 行声明一个垂直线性布局，其中有三个控件，一个 TextView 控件，一个 Button 控件，一个 ImageButton 控件。

② 第 12 行至第 15 行声明一个按钮，其 ID 为 Button01。

③ 第 18 行至第 22 行声明一个图片按钮，其 ID 为 ImageButton01。

【运行结果】

在 Eclipse 中启动模拟器，然后运行项目 ButtonExample，Button 和 ImageButton 控件举例运行结果如图 4.11 所示。

4.3.4 ImageView

ImageView（图片视图）用于显示图片。

图片来源的途径有 Drawable 下的图片对象，资源文件的 id，Content Provider 中的 URI 等。

ImageView 常用属性如表 4.8 所示。

表 4.8 ImageView 常用属性

属　　性	说　　明
android：src	设置 ImageView 要显示的图片源
android：scaleType	调整或移动图片来适应 ImageView 的尺寸 ImageView

ImageView 常用方法如表 4.9 所示。

表 4.9 ImageView 常用方法

属　　性	说　　明
setAlpha()	设置 ImageView 的透明度
setSelected()	设置 ImageView 的选中状态
setImageDrawable()	设置 ImageView 的内容为指定的 Drawable 对象
setImageResource()	设置 ImageView 的内容为指定 id 的资源
setImageBitmap()	设置 ImageView 的内容为指定的 Bitmap 对象
setImageURI()	设置 ImageView 的内容为指定的 URI

【例 4.11】 ImageView 举例。

【解题思路】

创建一个 TextView 控件和一个 ImageView 控件。

【开发步骤和程序分析】

(1) 在 Eclipse 中创建一个 ImageViewExample 应用项目，包名为 com.application.imageviewexample。

(2) 在 res/layout 目录下的 main.xml 文件中，设置一个水平居中的 TextView 控件和一个水平居中的 ImageView 控件。

在该文件中编辑代码如下：

```
<?xml version = "1.0" encoding = "utf - 8"?>
```

```
2  <LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
3            android:orientation = "vertical"
4            android:layout_width = "fill_parent"
5            android:layout_height = "fill_parent"
6            android:gravity = "center" >
7
8            <! -- 设置一个 TextView 控件 -- >
9            < TextView
10               android:layout_width = "wrap_content"
11               android:layout_height = "wrap_content"
12               android:textSize = "20sp"
13               android:textColor = "#00f"
14               android:text = "显示花卉图片\n"  />
15
16           <! -- 设置一个 ImageView 控件 -- >
17           < ImageView
18               android:id = "@ + id/IV_img"
19               android:layout_width = "match_parent"
20               android:layout_height = "wrap_content"
21               android:src = "@drawable/flower"
22               android:scaleType = "centerInside" />"
23
24 </LinearLayout >
```

① 第 9 行至第 14 行设置一个 TextView 控件，用于显示文本信息。

② 第 17 行至第 22 行设置一个 ImageView 控件，第 21 行设置该控件的图片来源于 drawable 目录中的 flower.jpg 文件。

【运行结果】

在 Eclipse 中启动模拟器，然后运行项目 ImageViewExample，运行结果如图 4.12 所示。



图 4.12　ImageView 控件

4.3.5　Checkbox 和 RadioButton

CheckBox（复选框）与 RadioButton（单选按钮）是经常用到的选择按钮，CheckBox 是同时可以选择多个选项的控件，而 RadioButton 是仅可以选择一个选项的控件，它们都继承自 CompoundButton。

一组单选按钮需要编制到一个 RadioGroup 中。

CheckBox 与 RadioButton 的常用方法如表 4.10 所示。

表 4.10　CheckBox 与 RadioButton 常用方法

方　　法	说　　明
isChecked()	判断控件是否为选中状态,选中则返回 True
setChecked()	通过参数传入,设置控件是否为选中状态
performClick()	调用 OnClickListener 监听器,即模拟一次单击操作
setOnCheckedChangeListener()	为控件设置 OnCheckedChangeListener 监听器

【例 4.12】　Checkbox 和 RadioButton 控件举例。

【解题思路】

创建 Checkbox 和 RadioButton 控件,有两个复选框和两个单选按钮。

【开发步骤和程序分析】

(1) 在 Eclipse 中创建一个 CheckboxRadioButtonExample 应用项目,包名为 com.application.checkboxradiobuttonexample。

(2) 在 res/layout 目录下的 main.xml 文件中,设置两个复选框和两个单选按钮。

在该文件中编辑代码如下:

```
<?xml version = "1.0" encoding = "utf - 8"?>
<! -- 声明一个垂直线性布局 -->
< LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = "vertical"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent">
      < TextView android:id = "@ + id/TextView01"
          android:layout_width = "fill_parent"
          android:layout_height = "wrap_content"
          android:text = "@string/hello"/>
      <! -- 声明两个 CheckBox 控件 -->
      < CheckBox android:id = "@ + id/CheckBox01"
          android:layout_width = "wrap_content"
          android:layout_height = "wrap_content"
          android:text = "音乐" >
      </CheckBox >
      < CheckBox android:id = "@ + id/CheckBox02"
          android:layout_width = "wrap_content"
          android:layout_height = "wrap_content"
          android:text = "舞蹈" >
      </CheckBox >
      <! -- 声明一个 RadioGroup 控件,其中定义了两个 RadioButton 控件 -->
      < RadioGroup android:id = "@ + id/RadioGroup01"
          android:layout_width = "wrap_content"
          android:layout_height = "wrap_content">
          < RadioButton android:id = "@ + id/RadioButton01"
```

```
27                    android:layout_width = "wrap_content"
28                    android:layout_height = "wrap_content"
29                    android:text = "男" >
30                </RadioButton>
31                <RadioButton android:id = "@ + id/RadioButton02"
32                    android:layout_width = "wrap_content"
33                    android:layout_height = "wrap_content"
34                    android:text = "女" >
35                </RadioButton>
36            </RadioGroup>
37 </LinearLayout>
```

① 第 2 行至第 37 行声明一个垂直线性布局,其中有一个 TextView 控件,两个 CheckBox 控件,在一个 RadioGroup 控件中,有两个 RadioButton 控件。

② 第 12 行至第 21 行声明两个 CheckBox 控件。

③ 第 23 行至第 36 行声明一个 RadioGroup 控件,其中定义了两个 RadioButton 控件。

【运行结果】

在 Eclipse 中启动模拟器,然后运行项目 CheckboxRadioButtonExample,运行结果如图 4.13 所示。

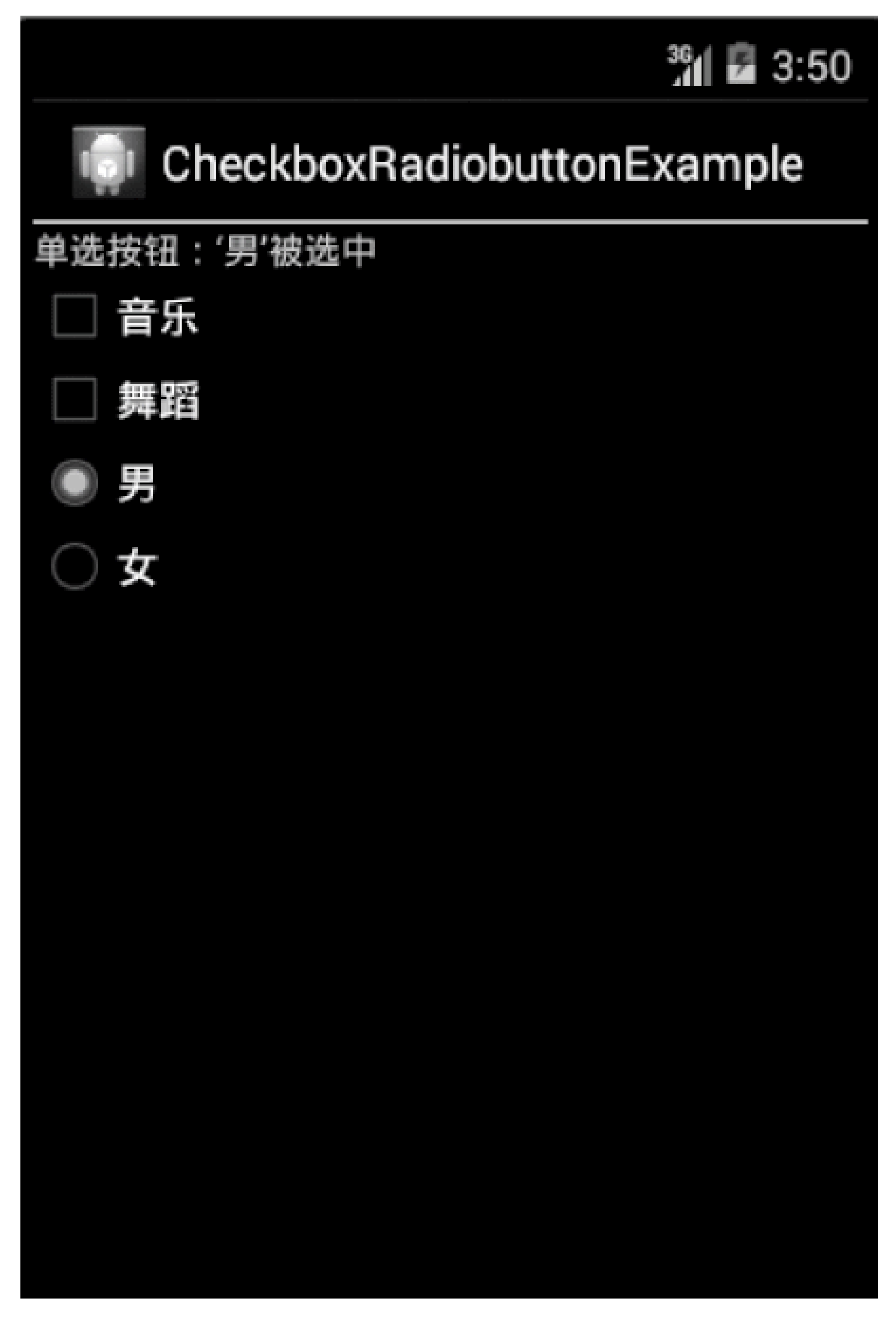


图 4.13　Checkbox 和 RadioButton 控件

4.3.6　AnalogClock 和 DigitalClock

AnalogClock(模拟时钟)控件用模拟时钟指针形式显示时间,模拟时钟只有时针和

分针。

DigitalClock(数字时钟)控件用数字形式显示时间,并精确到秒。

AnalogClock 控件和 DigitalClock 控件都在 android. widget 包下。

【例 4.13】 AnalogClock 控件和 DigitalClock 控件举例。

【解题思路】

创建 AnalogClock 和 DigitalClock 控件,有 2 个模拟时钟和 1 个数字时钟。

【开发步骤和程序分析】

(1) 在 Eclipse 中创建一个 AnalogClockDigitalClockExample 应用项目,包名为 com. application. analogclockdigitalclockexample。

(2) 在 res/layout 目录下的 main. xml 文件中,设置两个模拟时钟和一个数字时钟。

在该文件中编辑代码如下:

```
1  <?xml version = "1.0" encoding = "utf - 8"?>
2  <! -- 声明一个垂直线性布局 -->
3  < LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
4          android:orientation = "vertical"
5          android:layout_width = "fill_parent"
6          android:layout_height = "fill_parent"
7          android:gravity = "center_horizontal"
8          >
9      <! -- 定义一个数字时钟 -->
10     < DigitalClock
11             android:layout_width = "wrap_content"
12             android:layout_height = "wrap_content"
13             android:textSize = "12pt"
14             android:textColor = " # 0f0"
15         />
16     <! -- 定义一个模拟时钟 -->
17     < AnalogClock
18         android:layout_width = "wrap_content"
19         android:layout_height = "wrap_content"
20         />
21     <! -- 定义一个模拟时钟 -->
22     < AnalogClock
23             android:layout_width = "wrap_content"
24             android:layout_height = "wrap_content"
25             android:dial = "@drawable/watch"
26             android:hand_minute = "@drawable/hand"
27         />
28 </LinearLayout >
```

① 第 2 行至第 28 行声明一个垂直线性布局,其中有一个 DigitalClock 控件,两个 AnalogClock 控件。

② 第 10 行至第 15 行定义一个数字时钟,第 13 行定义字号为 12pt,第 14 行定义颜色

为绿色。

③ 第 17 行至第 20 行定义一个模拟时钟。

④ 第 22 行至第 27 行定义一个模拟时钟，第 25 行定义表盘图片，第 26 行定义时针图片。

【运行结果】

在 Eclipse 中启动模拟器，然后运行项目 AnalogClockDigitalClockExample，运行结果如图 4.14 所示。



图 4.14　AnalogClock 和 DigitalClock 控件

4.3.7　DatePicker 和 TimePicker

DatePicker 用于向用户提供年、月、日的日期数据，并允许用户进行修改。如果要修改日期数据，需要添加 onDateChangedListener 监听器。

DatePicker 常用方法如表 4.11 所示。

表 4.11　DatePicker 常用方法

方　　法	说　　明
getDayOfMonth()	获取日期天数
getMonth()	获取日期月份
getYear()	获取日期年份
updateDate()	根据传入的参数更新控件的日期值

TimePicker 用于向用户提供一天中的时间，可以是 24 小时制，也可以为 AM/PM 制，并允许用户进行修改。如果要修改时间数据，需要添加 onTimeChangedListener 监听器。

TimePicker 常用方法如表 4.12 所示。

表 4.12　TimePicker 常用方法

方　　法	说　　明
getCurrentHour()	获取当前小时
getCurrentMinute()	获取当前分钟
setCurrentHour ()	根据传入参数设置当前小时
is24HourView()	判断是否为 24 小时制

【例 4.14】 DatePicker 控件和 TimePicker 控件举例。

【解题思路】

创建 DatePicker 控件和 TimePicker 控件，有一个日期拾取器和一个时间拾取器。

【开发步骤和程序分析】

(1) 在 Eclipse 中创建一个 DatePickerTimePickerExample 应用项目，包名为 com.application.datepickertimepickerexample。

(2) 在 res/layout 目录下的 main.xml 文件中，设置一个日期拾取器和一个时间拾取器。

在该文件中编辑代码如下：

```
1  <?xml version = "1.0" encoding = "utf - 8"?>
2  <! -- 声明一个垂直线性布局 -- >
3  < LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
4       android:orientation = "vertical"
5       android:layout_width = "fill_parent"
6       android:layout_height = "fill_parent"
7       >
8       < TextView
9           android:layout_width = "fill_parent"
10          android:layout_height = "wrap_content"
11          android:text = "@string/hello"
12          />
13      <! -- 定义一个 DatePicker 控件 -- >
14      < DatePicker android:id = "@ + id/datePicker1"
15              android:layout_width = "wrap_content"
16              android:layout_height = "wrap_content"/>
17      <! -- 定义一个 TimePicker 控件 -- >
18      < TimePicker android:id = "@ + id/timePicker1"
19              android:layout_width = "wrap_content"
20              android:layout_height = "wrap_content"/>
21 </LinearLayout >
```

① 第 2 行至第 21 行声明一个垂直线性布局，其中有一个 TextView 控件、一个 DatePicker 控件、一个 TimePicker 控件。

② 第 14 行至第 16 行定义一个 DatePicker 控件。

③ 第 18 行至第 20 行定义一个 TimePicker 控件。

【运行结果】

在 Eclipse 中启动模拟器，然后运行项目 DatePickerTimePickerExample，运行结果如图 4.15 所示。



图 4.15　DatePicker 和 TimePicker 控件

4.3.8　用户登录界面设计

为了更好理解和综合应用基本控件，下面介绍综合实例：用户登录界面设计。

【例 4.15】 用户登录界面设计。

【解题思路】

一个用户登录界面包括的要素有：应用项目名称、用户账号和密码输入、“登录”和“注册”按钮。

应用项目名称使用 TextView 控件显示。用户账号和密码输入使用 TextView 控件提示，使用 EditText 控件进行账号和密码输入。“登录”和“注册”按钮使用 Button 控件。

为了界面美观，配上合适的背景图。

为统一界面风格，各个文本的字体、大小、颜色、边框等，编写样式文件 styles.xml。

为统一文字表述，编写字符串资源文件 styles.xml。

【界面设计】

(1) 画出“用户登录”界面草图。

本书使用 Android 模拟器为 320×480 像素，按此比例画出“用户登录”界面草图，如图 4.16 所示。



图 4.16 “用户登录”界面草图

其中，矩形框表示线性布局，圆角矩形表示控件。

(2) 由草图得界面设计方案如下：

① 一个外层 LinearLayout

该 LinearLayout 为垂直线性布局，设置背景图，包含一个 TextView 控件显示的标题，三个 LinearLayout 内层嵌套。

② “用户登录”标题 TextView 控件

③ 账号信息内层 LinearLayout

该内层 LinearLayout 为水平线性布局，一个 TextView 控件显示账号，左对齐屏幕，一个 EditText 控件用于输入账号。

④ 密码信息内层 LinearLayout

该内层 LinearLayout 为水平线性布局，一个 TextView 控件显示密码，左对齐屏幕，一个 EditText 控件用于输入密码，设置 android. password 为 True。

⑤ 按钮内层 LinearLayout

该内层 LinearLayout 为水平线性布局，两个 Button 控件为右对齐方式。

【开发步骤和程序分析】

(1) 在 Eclipse 中创建一个 UserLogin 应用项目，包名为 com. application. userlogin。

(2) 在 res/values 目录下的 string. xml 文件中，编辑代码如下：

```
<?xml version = "1.0" encoding = "utf - 8"?>
< resources >
    < string name = "hello"> UserLogin! </string >
    < string name = "app_name"> UserLogin </string >
    < string name = "userlogin">用户登录</string >
    < string name = "uid">用户名: </string >
    < string name = "upwd">密 码: </string >
    < string name = "ulogin">登录</string >
    < string name = "ureg">注册</string >
</resources >
```

(3) 在 res/values 目录下的 styles. xml 文件中,编辑代码如下:

```
<?xml version = "1.0" encoding = "utf - 8"?>
< resources >
      < style name = "title">
          < item name = "android:textSize"> 32sp </item >
          < item name = "android:textColor"># f37301 </item >
          < item name = "android:textStyle"> bold </item >
      </style >
      < style name = "text">
          < item name = "android:textSize"> 18sp </item >
          < item name = "android:textColor"># f37301 </item >
          < item name = "android:textStyle"> bold </item >
      </style >
      < style name = "button">
          < item name = "android:textSize"> 20sp </item >
          < item name = "android:textColor"># f37301 </item >
          < item name = "android:textStyle"> bold </item >
      </style >
</resources >
```

(4) 将背景图片文件 flower. jpg 复制到 res/drawable-mdpi 目录下。

(5) 在 res/layout 目录下的 main. xml 文件中,在设置的外层垂直分布的线性布局中,包含一个 TextView 控件用于显示标题,三个 LinearLayout 内层线性布局,分别用于输入账号、输入密码和显示按钮。

在该文件中编辑代码如下:

```
<?xml version = "1.0" encoding = "utf - 8"?>

<! -- 声明外层垂直分布的线性布局 -->
< LinearLayout
      xmlns:android = "http://schemas.android.com/apk/res/android"
      android:orientation = "vertical"
      android:layout_width = "fill_parent"
      android:layout_height = "fill_parent"
```

```
    android:gravity="center_horizontal"
    android:background="@drawable/flower"
    >
<TextView
      android:id="@+id/TextView0"
      android:text="@string/userlogin"
      style="@style/title"
      android:paddingTop="20dip"
      android:paddingBottom="20dip"
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:gravity="center"
      android:password="false"
     >
</TextView>

    <!-- 声明内层水平分布的线性布局,用于账号信息输入 -->
    <LinearLayout
        android:orientation="horizontal"
        android:paddingTop="25px"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"
        >
        <TextView
            android:text="@string/uid"
            android:layout_width="100px"
            style="@style/text"
            android:layout_height="wrap_content"
            android:layout_gravity="center_vertical"
        />
        <EditText
            android:id="@+id/etUid"
            android:singleLine="true"
            android:layout_width="150px"
            android:layout_height="wrap_content"
        />
    </LinearLayout>

    <!-- 声明内层水平分布的线性布局,用于密码信息输入 -->
    <LinearLayout
        android:orientation="horizontal"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"
```

```
        >
        <TextView
            android:text = "@string/upwd"
            style = "@style/text"
            android:layout_width = "100px"
            android:layout_height = "wrap_content"
            android:layout_gravity = "center_vertical"
        />
        <EditText
            android:id = "@ + id/etPwd"
            android:singleLine = "true"
            android:password = "true"
            android:layout_width = "150px"
            android:layout_height = "wrap_content"
        />
    </LinearLayout>

    <! -- 声明内层水平分布的线性布局,用于显示按钮 -->
    <LinearLayout
     android:paddingTop = "20dip"
     android:paddingBottom = "20dip"
     android:paddingRight = "20dip"
        android:orientation = "horizontal"
        android:gravity = "right"
        android:layout_width = "fill_parent"
        android:layout_height = "wrap_content"
        >
        <Button
            android:id = "@ + id/ulogin"
            android:text = "@string/ulogin"
            style = "@style/button"
            android:minWidth = "70dip"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            />
        <Button
            android:id = "@ + id/ureg"
            android:text = "@string/ureg "
            style = "@style/button"
            android:minWidth = "70dip"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            />
    </LinearLayout>

```

```
99 </LinearLayout>
```

① 第 4 行至第 99 行声明外层垂直分布的线性布局，包含一个 TextView 控件，三个 LinearLayout 内层嵌套，第 10 行设置背景图，背景图片取自 res/drawable-mpdi 目录中的 flower.jpg 设置文件。

② 第 12 行至第 23 行声明一个 TextView 控件，用于显示“用户登录”标题，第 14 行设置控件显示的内容，取自字符串资源文件 string.xml 中名为 userlogin 的内容，第 15 行设置控件文本显示的风格，其文本风格由样式文件 style.xml 中的 title 样式定义，第 16 行和第 17 行设置该控件与其相邻上、下控件间的间距为 20 像素。

③ 第 26 行至第 46 行声明内层水平分布的线性布局，用于账号信息输入。第 33 行至第 39 行声明一个 TextView 控件，第 34 行设置控件显示的内容，取自字符串资源文件 string.xml 中名为 tvUid 的内容，第 36 行设置控件文本显示的风格，其文本风格由样式文件 style.xml 中的 text 样式定义。第 40 行至第 45 行声明一个 EditText 控件，第 42 行设置该控件为单行编辑框。

④ 第 49 行至第 69 行声明内层水平分布的线性布局，用于密码信息输入。第 55 行至第 61 行声明一个 TextView 控件，第 62 行至第 68 行声明一个 EditText 控件，第 65 行设置该 EditText 的 password 为 True 编辑框。

⑤ 第 72 行至第 97 行声明内层水平分布的线性布局，用于显示按钮。第 81 行至第 88 行和第 89 行至第 96 行分别声明两个 Button 控件，第 85 行和第 93 行分别设置这两个控件的最小宽度为 70 像素，以确保按钮文本显示完整。



图 4.17　用户登录界面

【运行结果】

在 Eclipse 中启动模拟器，然后运行项目 UserLogin，运行结果如图 4.17 所示。

4.4 小　　结

本章主要介绍了以下内容：

(1) 用户界面(User Interface，UI)是系统和用户之间进行信息交换的媒介，实现信息的内部形式与人类可以接受形式之间的转换。当前流行的是图形用户界面(Graphical User Interface，GUI)。

View(视图)是所有可视化窗体控件的基类，所有在界面上的可见元素都是 View 的子类。控件是 Android 用户界面中的组成元素，控件的父类是 View。ViewGroup(视图组)是 android. view. Viewgroup 的一个实例，是一种特殊类型的视图，可以充当 View 的容器。

(2) Layout(布局)是 ViewGroup 类的子类，为视图控件提供排列结构。Android 常用的布局方式有：线性布局(LinearLayout)、表格布局(TableLayout)、帧布局(FrameLayout)、网格布局(GridLayout)、相对布局(RelativeLayout)、绝对布局(AbsoluteLayout)。

线性布局是最常用的布局方式。线性布局的控件定义在< LinearLayout ></LinearLayout >标签之间，它将其包含的控件元素按水平或者垂直方向顺序排列。

表格布局通过指定行和列将界面元素添加到表格中，以多行多列的方式显示子对象，但它不会显示行、列或单元格的边框线。

帧布局为加入其中的每个控件创建一个空白区域，该空白区域称为一帧，每个控件占据一帧。

网格布局是 Android SDK 4.0 新增加的布局方式，它将用户界面划分为网格，界面元素可随意摆放在网格中，网格布局比表格布局在界面设计上更加灵活，在网格布局中界面元素可以占用多个网格。

相对布局是一种灵活的布局方式，能够通过指定界面元素与其他元素的相对位置关系，确定界面中所有元素的布局位置，从而保证在各种屏幕尺寸的手机上正确显示界面布局。

绝对布局中所有控件的位置通过设置控件的坐标来指定。

(3) Android 的可视控件都在 android. widget 包内。Widget 常用的控件包括：TextView、EditText、Button、ImageButton、ImageView、Checkbox、RadioButton、AnalogClock、DigitalClock、DatePicker、TimePicker 等。

TextView(文本框)用于向用户显示文本内容，其显示的文本只能在初始设置时或在程序中修改。

EditText(编辑框)用于显示文本的内容，并可对文本进行编辑，而 TextView 仅用于设置显示的文本。EditText 是一个具有编辑功能的 TextView，EditText 是 TextView 的子类。

Button(按钮)是广泛使用的控件，它继承自 TextView。Button 上可显示文本，它的背景可以改变颜色或显示图片。

ImageButton(图片按钮)控件和 Button 控件功能一致，但显示效果不同，在

ImageButton 的按钮中只显示图片，不显示文本。

ImageView(图片视图)用于显示图片，图片来源的途径有 Drawable 下的图片对象，资源文件的 id，Content Provider 中的 URI 等。

CheckBox(复选框)与 RadioButton(单选按钮)是经常用到的选择按钮，CheckBox 是同时可以选择多个选项的控件，而 RadioButton 是仅可以选择一个选项的控件，它们都继承自 CompoundButton。

AnalogClock(模拟时钟)控件用模拟时钟指针形式显示时间，模拟时钟只有时针和分针。DigitalClock(数字时钟)控件用数字形式显示时间，并精确到秒。

DatePicker 用于向用户提供年、月、日的日期数据，并允许用户进行修改。TimePicker 用于向用户提供一天中的时间，可以是 24 小时制，也可以为 AM/PM 制，并允许用户进行修改。

习　题　4

一、选择题

4.1　通过指定行和列将界面元素添加到表格中的布局方式是________。

A. 线性布局　　B. 表格布局　　C. 帧布局　　D. 网格布局

4.2　将其包含的控件元素按水平或者垂直方向顺序排列的布局方式是________。

A. 相对布局　　B. 绝对布局　　C. 线性布局　　D. 帧布局

4.3　通过指定界面元素与其他元素的相对位置关系，确定界面中所有元素的布局位置的布局方式是________。

A. 相对布局　　B. 表格布局　　C. 绝对布局　　D. 网格布局

4.4　将用户界面划分为网格，界面元素可随意摆放在网格中的布局方式是________。

A. 线性布局　　B. 帧布局　　C. 表格布局　　D. 网格布局

4.5　用于显示文本的内容，并可对文本进行编辑的控件是________。

A. TextView　　B. EditText　　C. Button　　D. AnalogClock

4.6　同时可以选择多个选项的控件是________。

A. Checkbox　　B. RadioButton　　C. Button　　D. AnalogClock

4.7　用于向用户显示文本内容的控件是________。

A. TextView　　B. EditText　　C. DigitalClock　　D. DatePicker

4.8　可显示文本，可以改变颜色或显示图片的按钮控件是________。

A. ImageButton　　B. RadioButton　　C. Button　　D. AnalogClock

二、填空题

4.9　用户界面(User Interface，UI)是系统和用户之间进行________的媒介。

4.10　View(视图)是所有________的基类。

4.11　Android 常用的布局方式有：线性布局，表格布局，帧布局，网格布局，________，绝对布局。

4.12　网格布局比表格布局在界面设计上更加灵活，在网格布局中的界面元素可以占

用________网格。

4.13　相对布局是一种灵活的布局方式，能够通过指定界面元素与其他元素的________关系，确定界面中所有元素的布局位置。

4.14　Android 的可视控件都在________包内。

4.15　EditText 是一个具有________的 TextView。

4.16　ImageButton 的按钮中只显示________，不显示文本。

4.17　ImageView 图片来源的途径有 Drawable 下的________，资源文件的 id 等。

4.18　模拟时钟只有时针和分针，而数字时钟可以精确到________。

三、问答题

4.19　什么是用户界面？什么是图形用户界面？

4.20　简述 View 和 ViewGroup 的概念。

4.21　Android 常用的布局方式有哪几种？各有何特点？

4.22　简述 Widget 常用的控件的种类和特点。

四、应用题

4.23　设计一个个人主页的登录界面，要求有“用户账号”“密码”和“确认密码”等输入编辑框，“登录”和“注册”两个按钮。

4.24　设计一个提交订单的用户界面，要求有“用户名”“地址”“电话”“商品名称”“费用小计”“配送方式”等输入编辑框，“提交订单”按钮。

4.25　设计一个用户界面，界面包括一组单选按钮，一个文本输入框和一个“确定”按钮。单选按钮的选项有“普通用户”“银卡用户”和“金卡用户”。

第 5 章　Android 事件处理、高级控件和菜单

本章要点

- 基于监听接口的事件处理模型有三个对象：Event Source（事件源）、Event（事件）和 Event Listener（事件监听器）。
- 基于回调机制的事件处理是将事件的处理绑定在控件上，由用户界面控件自己处理事件，回调机制需要自定义 View 来实现。
- 与适配器相关的控件有 AutoCompleteTextView（自动完成文本框）、Spinner（下拉列表）、Gallery（画廊视图）、ListView（列表视图）、GridView（网格视图）。
- 与视图相关的控件有 ScrollView（滚动视图）、TabHost（选项卡）、ImageSwitcher（图像切换器）。
- 进度条与滑块进度条有 ProgressBar（进度条）、SeekBar（拖动条）、RatingBar（星级评分条）。
- Android 平台下有三类菜单：选项菜单（OptionMenu）、子菜单（SubMenu）、上下文菜单（ContentMenu）。

Android 用户界面的各种控件能被用户看到，只有当它们与用户进行交互，触发了新的动作才活动起来。本章介绍 Android 事件处理机制、常用高级控件和菜单等内容。

5.1　Android 事件处理机制

Android 事件的处理机制有两种：一种是基于监听接口的事件处理，一种是基于回调机制的事件处理。

5.1.1　基于监听接口的事件处理

本节介绍基于监听接口的事件处理模型、监听器接口与回调方法和事件监听器接口的实现方法等内容。

1. 基于监听接口的事件处理模型

在基于监听接口的事件处理模型中，有三个对象：

(1) Event Source（事件源）。事件发生的场所，就是产生事件的各个控件，例如按钮、窗口、菜单等。

(2) Event（事件）。由用户的一次操作触发事件，事件封装了控件上发生的特定事情，一般通过 Event 对象获得。

(3) Event Listener(事件监听器)。监听事件源发生的事件,并对各种事件做出相应的处理给出响应。

事件监听处理流程如下:

(1) 将事件监听器注册到事件源,即为事件源添加事件监听器;

(2) 触发事件源上的事件;

(3) 生成事件对象;

(4) 事件被发送到事件监听器;

(5) 调用事件监听器中相应的事件处理方法来处理事件并给出响应。

事件监听处理流程如图 5.1 所示。

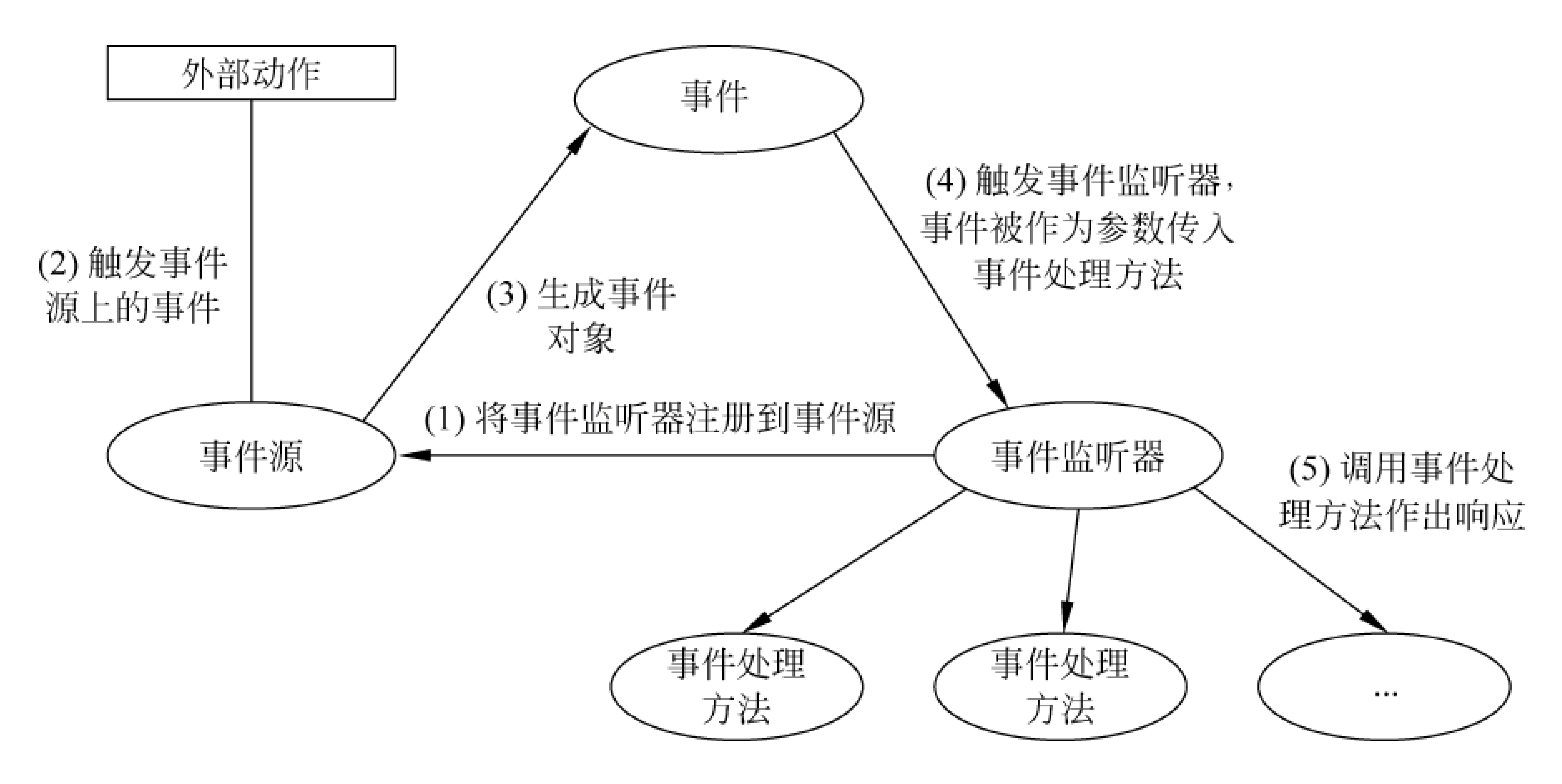


图 5.1　事件监听处理流程

通过下面的例题帮助我们理解事件监听处理流程。

【例 5.1】 事件监听处理举例。

【解题思路】

定义一个按钮作为事件源,定义一个单击事件的监听器,将事件监听器注册到事件源。当按钮被单击时,编辑框内出现"按钮已被单击!"

【开发步骤和程序分析】

(1) 在 Eclipse 中创建一个 EventListenerProcessing 应用项目,包名为 com.application.eventlistenerprocessing。

(2) 在 res/layout 目录下的 main.xml 文件中,设置一个 EditText 控件,一个 Button 控件。

在该文件中编辑代码如下:

```
<?xml version = "1.0" encoding = "utf - 8"?>
<LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
        android:orientation = "vertical"
        android:layout_width = "fill_parent"
        android:layout_height = "fill_parent"
        android:gravity = "left"
```

```
7        >
8     <!-- 设置一个 EditText 控件 -->
9     <EditText
10        android:id = "@ + id/edtext"
11        android:layout_width = "fill_parent"
12        android:layout_height = "wrap_content"
13        android:editable = "false"
14        android:cursorVisible = "false"
15        android:textSize = "10pt"
16        />
17     <!-- 设置一个 Button 控件,该按钮作为事件源 -->
18     <Button
19        android:id = "@ + id/btn"
20        android:layout_width = "wrap_content"
21        android:layout_height = "wrap_content"
22        android:text = "按钮"
23        />
24 </LinearLayout>
```

① 第 9 行至第 16 行定义 EditText 控件。

② 第 18 行至第 23 行定义 Button 控件,该按钮作为事件源。

(3) 在 src/com.application.eventlistenerprocessing 包下的 EventListenerProcessingActivity.java 文件中,为按钮绑定事件监听器,定义一个实现 View.OnClickListener 接口的实现类,该实现类作为事件监听器。

在该文件中编辑代码如下:

```
package com.application.eventlistenerprocessing;

import com.application.eventlistenerprocessing.R;
import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;

public class EventListenerProcessingActivity extends Activity {

      @Override
      public void onCreate(Bundle savedInstanceState) {
          super.onCreate(savedInstanceState);
          setContentView(R.layout.main);
          //获取 Button 控件的对象的 btn 按钮
          Button btn = (Button) findViewById(R.id.btn);
          //为该按钮绑定事件监听器
          btn.setOnClickListener(new MyClickListener());
```

```
20        }
21        //定义一个实现 View.OnClickListener 接口的实现类,该实现类作为事件监听器
22        class MyClickListener implements View.OnClickListener {
23            //实现上述接口的事件处理方法,当 btn 按钮被单击时,该事件处理方法被激发,
24            //编辑框内变为"按钮已被单击!"
25            @Override
26            public void onClick(View v) {
27                EditText edtext = (EditText) findViewById(R.id.edtext);
28                edtext.setText("按钮已被单击 !");
29            }
30        }
31 }
```

① 第 17 行获取 Button 控件的对象的 btn 按钮。

② 第 19 行为该按钮绑定事件监听器。

③ 第 22 行至第 30 行定义一个实现 View.OnClickListener 接口的实现类,该实现类作为事件监听器。其中,第 26 行至第 29 行是实现上述接口的事件处理方法,当 btn 按钮被单击时,该事件处理方法被激发,编辑框内变为“按钮已被单击!”。

【运行结果】

在 Eclipse 中启动模拟器,然后运行项目 EventListenerProcessing,在出现的界面中单击“按钮”时,编辑框内显示“按钮已被单击!”,运行结果如图 5.2 所示。



图 5.2　事件监听处理举例

事件处理有以下三个步骤：

（1）为事件源对象设置监听器（1 行代码）。

（2）系统将事件封装成事件对象。

（3）实现事件处理方法（多行代码）。

在上述步骤中，程序设计人员只需完成（1）、（3）两个步骤的工作。

2. 监听器接口与回调方法

1）OnClickListener 接口

（1）功能

该接口处理单击事件。在触摸模式下，View 对象被按下并抬起。在键盘模式下，View 对象获得焦点后，单击“确定”键或按下轨迹球。

（2）对应的回调方法

```
public void onClick(View v)
```

2）OnLongClickListener 接口

（1）功能

处理长按下事件，即处理长时间按下某个 View 时触发的事件。

（2）对应的回调方法

```
public boolean onLongClick(View v)
```

3）OnFocusChangeListener 接口

（1）功能

处理控件焦点发生改变事件。

（2）对应的回调方法

```
public void onFocusChange (View v, Boolean hasFocus)
```

4）OnKeyListener 接口

（1）功能

对手机键盘进行监听的接口。

（2）对应的回调方法

```
public boolean onKey (View v, int keyCode, KeyEvent event)
```

5）OnTouchListener 接口

（1）功能

处理手机屏幕事件的监听接口。

（2）对应的回调方法

```
public void onTouch (View v, MotionEvent event)
```

6）OnCreateContextMenuListener 接口

(1) 功能

处理上下文菜单显示事件的监听接口。

(2) 对应的回调方法

```
public void onCreateContextMenu(ContextMenu menu, View v, ContextMenuInfo info)
```

【例 5.2】 在编辑框中输入信息,单击按钮后将输入的信息显示在标题栏中。

【解题思路】

定义一个按钮,该按钮作为事件源,定义一个单击事件的监听器,将该按钮绑定到事件监听器,再定义一个编辑框。当用户在编辑框中输入信息后,单击按钮,输入的信息被送到标题栏中。

【开发步骤和程序分析】

(1) 在 Eclipse 中创建一个 EditButton 应用项目,包名为 com. application. editbutton。

(2) 在 res/layout 目录下的 btn. xml 文件中,设置一个 EditText 控件,一个 Button 控件。

在该文件中编辑代码如下:

```
1 <?xml version = "1.0" encoding = "utf - 8"?>
2 < LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
3      android:orientation = "vertical"
4      android:layout_width = "fill_parent"
5      android:layout_height = "fill_parent"
6      >
7     <! -- 设置一个 EditText 控件 -->
8     < EditText android:id = "@ + id/edit"
9           android:layout_width = "fill_parent"
10          android:layout_height = "wrap_content"
11          android:text = "" />
12    <! -- 设置一个 Button 控件,该按钮作为事件源 -->
13    < Button android:id = "@ + id/btn"
14          android:layout_width = "wrap_content"
15          android:layout_height = "wrap_content"
16          android:text = "获取编辑框的值" />
17 </LinearLayout >
```

① 第 2 行至第 17 行定义一个垂直线性布局。

② 第 8 行至第 11 行定义 EditText 控件。

③ 第 13 行至第 16 行定义 Button 控件,该按钮作为事件源。

(3) 在 src/ com. application. editbutton 包下的 EditButtonActivity. java 文件中,为按钮绑定事件监听器,定义一个实现 View. OnClickListener 接口的实现类,该实现类作为事件监听器。

在该文件中编辑代码如下:

```
package com.application.editbutton;
```

```
2  import com.application.editbutton.R;
3  import android.app.Activity;
4  import android.os.Bundle;
5  import android.view.View;
6  import android.widget.Button;
7  import android.widget.EditText;
8
9  public class EditButtonActivity extends Activity {
10
11       @Override
12       public void onCreate(Bundle savedInstanceState) {
13           super.onCreate(savedInstanceState);
14           setTitle("EditButton");
15           setContentView(R.layout.btn);
16           //获取 Button 控件的对象的 btn 按钮
17           Button btn = (Button) findViewById(R.id.btn);
18           //为该按钮绑定事件监听器
19           btn.setOnClickListener(new editbuttonlistener());
20       }
21       //定义一个实现 View.OnClickListener 接口的实现类,该实现类作为事件监听器
22       private class editbuttonlistener implements View.OnClickListener {
23           //实现上述接口的事件处理方法,当 btn 按钮被单击时,该事件处理方法被激发,
24           //将编辑框中输入信息传送到标题栏中
25           public void onClick(View v) {
26               EditText edit = (EditText) findViewById(R.id.edit);
27               CharSequence textvalue = edit.getText();
28               setTitle("您输入的值是: " + textvalue);
29           }
30       }
31 }
```

① 第 17 行,在 onCreate()方法中,获取按钮对象。

② 第 19 行为该按钮绑定事件监听器,该监听器的接口实现类是 editbuttonlistener。

③ 第 22 行至第 30 行,在 EditButtonActivity 类的内部定义一个实现 View.OnClickListener 接口的实现类,该实现类作为事件监听器。其中,第 25 行至第 29 行是实现上述接口的事件处理方法,第 27 行使用 getText()方法从 EditText 对象中取出字符串信息,并赋予到字符串变量 textvalue 之中,第 28 行使用 setTitle()方法将一个字符串信息传送到标题栏中,该信息为“您输入的值是:”连接 textvalue 中的值串。

【运行结果】

在 Eclipse 中启动模拟器,然后运行项目 LinearLayout,在界面的编辑框中输入“Success”,单击“获取编辑框的值”按钮后,在标题栏内显示“您输入的值是: Success”,运行结果如图 5.3 和图 5.4 所示。



图 5.3　获取编辑框中内容的界面

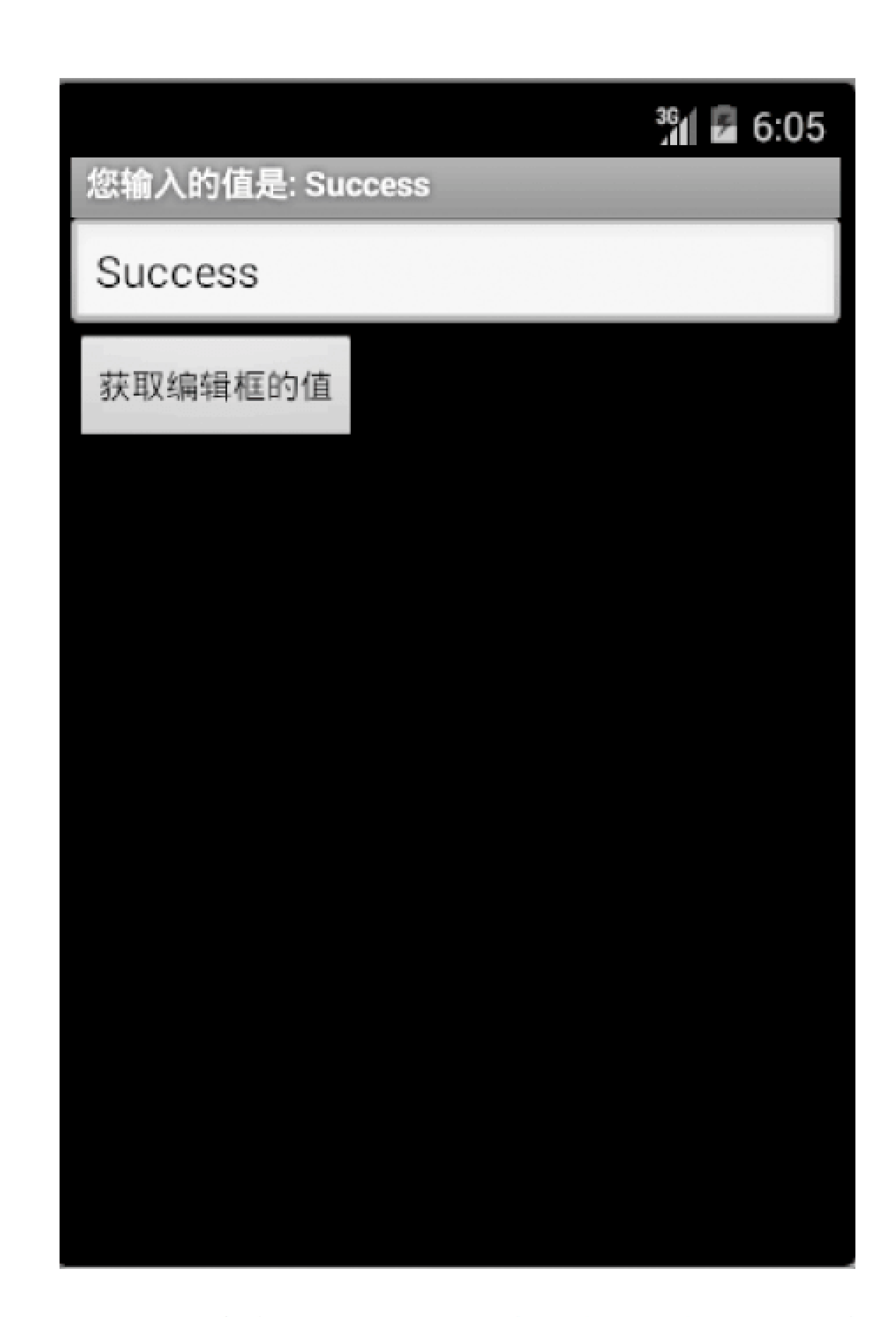


图 5.4　单击按钮后获取编辑框中输入的内容

【例 5.3】 对一组单选按钮中的选择项进行清除。

【解题思路】

定义一组单选按钮，再定义一个“清除”按钮，该按钮作为事件源，定义一个单击事件的监听器，将该按钮绑定到事件监听器。当用户单击“清除”按钮时，所有单选按钮的状态都变为非选择状态。

【开发步骤和程序分析】

(1) 在 Eclipse 中创建一个 RadioButton 应用项目，包名为 com. application. radiobutton。

(2) 在 res/layout 目录下的 main. xml 文件中，设置三个 RadioButton 控件，一个 Button 控件。

在该文件中编辑代码如下：

```
<?xml version = "1.0" encoding = "utf - 8"?>
<LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:orientation = "vertical">
    <!-- 设置一个 RadioGroup 控件 -->
    <RadioGroup
        android:id = "@ + id/menu"
        android:checkedButton = "@ + id/lunch"
```

```
10          android:layout_width = "fill_parent"
11          android:layout_height = "wrap_content"
12          android:orientation = "vertical"
13          >
14          <! -- 设置 RadioButton 控件 -->
15          < RadioButton
16              android:id = "@ + id/lunch "
17              android:text = "红色" />
18          < RadioButton
19              android:id = "@ + id/blue"
20             android:text = "蓝色" />
21         < RadioButton
22             android:id = "@ + id/green"
23             android:text = "绿色" />
24     </RadioGroup >
25     <! -- 设置一个 Button 控件,该按钮作为事件源 -->
26     < Button
27         android:id = "@ + id/clear"
28         android:layout_width = "wrap_content"
29         android:layout_height = "wrap_content"
30         android:text = "清除" />
31 </LinearLayout >
```

① 第 2 行至第 31 行定义一个垂直线性布局。

② 第 7 行至第 24 行定义一个 RadioGroup 控件,其中定义了三个 RadioButton,第 9 行设置 ID 为 lunch 的单选按钮为选中状态。

③ 第 26 行至第 30 行定义 Button 控件,该按钮作为事件源。

(3) 在 src/com. application. radiobutton 包下的 RadioButtonActivity. java 文件中,为按钮绑定事件监听器,定义一个实现 View. OnClickListener 接口的实现类,该实现类作为事件监听器。

在该文件中编辑代码如下:

```
package com.application.radiobutton;

import com.application.radiobutton.R;
import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.RadioGroup;

//定义一个实现 View.OnClickListener 接口的实现类,该实现类作为事件监听器
public class RadioButtonActivity extends Activity implements View.OnClickListener {
    private RadioGroup mRadioGroup;

```

```
14    @Override
15    protected void onCreate(Bundle savedInstanceState) {
16        super.onCreate(savedInstanceState);
17        setContentView(R.layout.main);
18        setTitle("RadioButton");
19        mRadioGroup = (RadioGroup) findViewById(R.id.menu);
20        //获取 Button 控件的对象的 clearButton 按钮
21        Button clearButton = (Button) findViewById(R.id.clear);
22        //为该按钮绑定事件监听器
23        clearButton.setOnClickListener(this);
24    }
25
26   //实现上述接口的事件处理方法,当 btn 按钮被单击时,该事件处理方法被激发,
27   //设置 RadioGroup 组内所有单选按钮都为未选中状态
28    @Override
29    public void onClick(View v) {
30        mRadioGroup.clearCheck();
31    }
32 }
```

① 第 11 行至第 32 行,定义 RadioButtonActivity 类,实现 View.OnClickListener 事件监听器接口。

② 第 14 行至第 24 行,重写 onCreate()方法,其中,第 19 行为 RadioGroup 对象 mRadioGroup 获取实例,第 21 行为 Button 对象 clearButton 获取实例,第 23 行为该 clearButton 按钮绑定事件监听器,this 指本类对象,即 RadioButtonActivity 类对象。

③ 第 28 行至第 31 行,重写 View.OnClickListener 接口的回调方法 onClick(),调用 RadioGroup 类的 clearCheck()方法,作用为设置 RadioGroup 组内所有单选按钮都为未选中状态。

【运行结果】

在 Eclipse 中启动模拟器,然后运行项目 RadioButton,在出现的界面中有三个单选按钮,其中,红色按钮被选中,当用户单击"清除"按钮时,所有单选按钮的状态都变为非选择状态,运行结果如图 5.5 所示。



图 5.5　清除单选按钮的选中状态

5.1.2　基于回调机制的事件处理

回调机制是将事件的处理绑定在控件上,由

用户界面控件自己处理事件。回调机制需要自定义 View 来实现。

每个 View 类都有自己的处理事件的回调方法，开发人员可以通过重写 View 中这些回调方法来实现需要响应的事件。

View 类提供了许多公用的捕获用户在界面上触发事件的方法。为了捕获和处理事件，必须继承某个类(如 View 类)，并重写这些方法，以便开发人员自己定义具体的处理逻辑代码。

下面介绍一些常见的回调方法。

1. onKeyDown()

该方法用于捕捉手机键盘按键按下的事件，其格式如下：

```
public boolean onKeyDown(int keyCode, KeyEvent event)
```

1）keyCode

该参数为 int 类型，表示被按下的键的键值(即键盘码)。

2）event

该参数为按键事件的对象，封装了触发事件的详细信息。

3）返回值

返回值是 boolean 类型，当返回 True 时，表示已完整地处理了该事件。

2. onKeyUp()

该方法用于捕捉手机键盘按键抬起的事件，其格式如下：

```
public boolean onKeyUp(int keyCode, KeyEvent event)
```

1）keyCode

该参数为 int 类型，表示被抬起的键的键值。

2）event

该参数为按键事件的对象，封装了触发事件的详细信息。

3）返回值

返回值是 boolean 类型，当返回 True 时，表示已完整地处理了该事件。

3. onTouchEvent()

该方法用于处理手机屏幕的触摸事件，其格式如下：

```
public boolean onTouchEvent(MotionEvent event)
```

参数 event 为手机屏幕触摸事件封装类的对象，封装了该事件的详细信息。

返回值是 boolean 类型，与键盘响应事件的返回值含义相同。

以下情形由 onTouchEvent 方法处理：

(1) 屏幕被按下。此时 getAction()的值为 MotionEvent.ACTION_DOWN。

(2) 屏幕被抬起。此时 getAction()的值为 MotionEvent.ACTION_UP。

(3) 在屏幕中拖动。此时 getAction()的值为 MotionEvent.ACTION_MOVE。

4. onTrackballEvent()

该方法用于处理手机中轨迹球事件，其格式如下：

```
public boolean onTrackballEvent(MotionEvent event)
```

参数 event 为手机轨迹球事件封装类的对象，封装了该事件的详细信息。

其参数与返回值和上面的方法相同。

使用轨迹球的特点是：使用简单，比键盘操作表示状态的数据更细化。

在模拟器运行状态下，F6 键打开模拟器轨迹球，然后用鼠标移动模拟轨迹球事件。

5. onFocusChanged ()

该方法用于处理焦点改变的事件，其格式如下：

```
Protected void onFocusChanged(Boolean gainFocus, int direction, Rect previouslyFocusedRect)
```

1) gainFocus

该参数是 boolean 类型，表示触发该事件的 View 是否获得了焦点。

2) direction

该参数是 int 类型，表示焦点移动方向。

3) previouslyFocusedRect

该参数是 Rect 类型，表示触发事件时，前一个获得焦点的矩形区域，即表示焦点从哪里来。

与焦点有关的常用方法如表 5.1 所示。

表 5.1　与焦点有关的常用方法

方　　法	说　　明
setFocusable	设置控件是否可以拥有焦点
isFocusable	监测控件是否可以拥有焦点
setNextFocusDownId	设置焦点向下移动后获得焦点控件的 ID
hasFocus	返回了控件的父控件是否获得了焦点
requestFocus	试图获得焦点
isFocusableTouchMode	在触摸模式下，设置控件是否可以拥有焦点。默认情况下是不能的

【例 5.4】　在手机屏幕区域内触摸滑动，当按下、抬起和滑动时获取触点的坐标、压力和大小等信息。

【解题思路】

使用基于回调机制的事件处理，以获取按下、抬起和滑动时获取触点的坐标、压力和大小等信息。

【开发步骤和程序分析】

(1) 在 Eclipse 中创建一个 CallbackMechanism 应用项目，包名为 com. application. callbackmechanism。

(2) 在 res/layout 目录下的 main. xml 文件中，设置两个 TextView 控件。

在该文件中编辑代码如下：

```
<?xml version = "1.0" encoding = "utf - 8"?>
<LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = "vertical"
    android:background = "#AA00DD"
    android:layout_width = "fill_parent"
```

```
6      android:layout_height = "fill_parent">
7      <!-- 设置一个 TextView 控件,其资源 ID 为 touch_area -->
8      < TextView
9          android:id = "@ + id/touch_area"
10         android:layout_width = "fill_parent"
11         android:layout_height = "360dip"
12         android:text = "触摸事件测试范围"
13         android:textColor = "＃99FFFF"
14         />
15     <!-- 设置一个 TextView 控件,其资源 ID 为 event_label -->
16     < TextView
17         android:id = "@ + id/event_label"
18         android:layout_width = "fill_parent"
19         android:layout_height = "wrap_content"
20         android:text = "触摸事件: "
21         android:textColor = "＃FFFFFF"
22       />
23 </LinearLayout >
```

① 第 2 行至第 23 行定义一个垂直线性布局。

② 第 8 行至第 14 行定义一个 TextView 控件,其资源 ID 为 touch_area;第 10 行设置 TextView 高为 360dip,以使触摸滑动信息显示在屏幕下方。

③ 第 16 行至第 22 行定义另一个 TextView 控件,其资源 ID 为 event_label。

(3) 在 src/com. application. callbackmechanism 包下的 CallbackMechanismActivity. java 文件中,重写 onTouchEvent()方法。该方法用于处理手机屏幕的触摸事件,依据事件的不同类型进行处理。

在该文件中编辑代码如下:

```
package com.application.callbackmechanism;

import com.application.callbackmechanism.R;
import android.app.Activity;
import android.os.Bundle;
import android.view.MotionEvent;
import android.widget.TextView;

public class CallbackMechanismActivity extends Activity {
    private TextView eventlable;

    //重写 onCreate()方法
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
     //eventlable 对象实例化,将资源 ID 为 event_label 的 TextView 对象赋予变量 eventlable 中
```

```
18          eventlable = (TextView) findViewById(R.id.event_label);
19      }
20
21      //重写 onTouchEvent()方法,该方法用于处理手机屏幕的触摸事件
22      @Override
23      public boolean onTouchEvent(MotionEvent event) {
24          int action = event.getAction();
25          //开关语句,根据 action 中不同值,将不同值传入到 Display()方法中
26          switch (action) {
27          case (MotionEvent.ACTION_DOWN):
28              Display("ACTION_DOWN", event);
29              break;
30          case (MotionEvent.ACTION_UP):
31              Display("ACTION_UP", event);
32              break;
33          case (MotionEvent.ACTION_MOVE):
34              Display("ACTION_MOVE", event);
35          }
36          return super.on TouchEvent(event);
37      }
38
39      //定义 Display()方法
40      public void Display(String eventType, MotionEvent event) {
41          int x = (int) event.getX();
42          int y = (int) event.getY();
43          float pressure = event.getPressure();
44          float size = event.getSize();
45          String msg = "";
46          msg += "事件类型:" + eventType + "\n";
47          msg += "坐标(x , y):" + String.valueOf(x) + " , " + String.valueOf(y) + "\n";
48          msg += "触点压力:" + String.valueOf(pressure) + "\n";
49          msg += "触点尺寸:" + String.valueOf(size) + "\n";
50          eventlable.setText(msg);
51      }
52  }
```

① 第 6 行,引入 android.view.MotionEvent 类,在代码中要使用触摸滑动类的对象。

② 第 7 行,引入 android.widget.TextView 类,在代码中要给一个 TextView 类的对象赋值。

③ 第 10 行,声明一个 TextView 类的对象,名为 eventlable。

④ 第 13 行至第 19 行,重写 onCreate()方法;第 18 行为 eventlable 对象实例化,即将资源 ID 为 event_label 的 TextView 对象赋予变量 eventlable 中。

⑤ 第 22 行至第 37 行,重写 onTouchEvent()方法,该方法用于处理手机屏幕的触摸事件。其中,event 参数是一个 MotionEvent 对象,在第 24 行,通过 getAction()方法获取事件

的状态，并返回结果到整型变量 action 中。

⑥ 第 26 行至第 35 行，是一个开关语句，根据 action 中不同值，将不同值传入到 Display()方法中。当 action 值为 MotionEvent. ACTION_DOWN 时，即按下的时候，调用方法 Display("ACTION_DOWN", event)；当 action 值为 MotionEvent. ACTION_UP 时，即抬起的时候，调用方法 Display("ACTION_UP", event)；当 action 值为 MotionEvent. ACTION_MOVE 时，即触摸的时候，调用方法 Display("ACTION_MOVE", event)。

⑦ 第 40 行至第 51 行，定义 Display()方法，通过 onTouchEvent()方法的调用，在该方法中获取触摸屏幕事件的状态，包括触点坐标、触屏压力、触点尺寸等信息；第 41 行至第 42 行获取触点相对坐标的信息，第 43 行获取触屏压力大小，第 44 行获取触点尺寸。

【运行结果】

在 Eclipse 中启动模拟器，然后运行项目 CallbackMechanism，运行结果如图 5.6 所示。

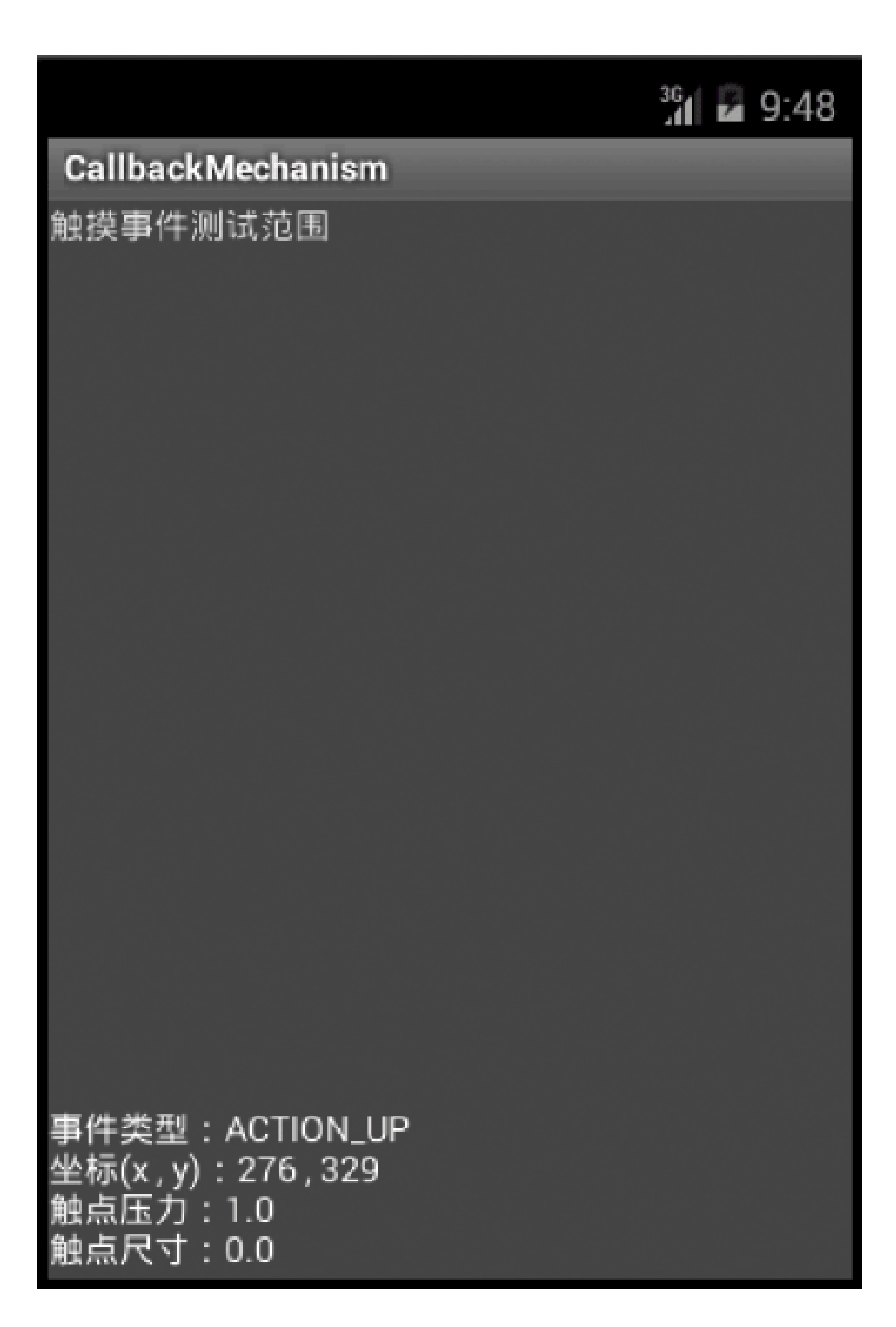


图 5.6　手机屏幕内触摸滑动及相关信息

5.2　Android 常用高级控件

在 Android 中，Widget 常用高级控件有：

(1) 与适配器相关的控件。AutoCompleteTextView、Spinner、GridView、Gallery、ListView。

(2) 视图控件。ScrollView、TabHost、ImageSwitcher。

(3) 滑块与进度条。ProgressBar、SeekBar、RatingBar。

Adapter(适配器)用于将数据绑定到用户界面上,常用的适配器有 ArrayAdapter、SimpleAdapter、SimpleCursorAdapter,它们都是继承 BaseAdapter,Adapter 位于 android.widget 包下。

- ArrayAdapter。简单易用的 Adapter,通常用于将数组或 List 集合的值包装成多个列表项。
- SimpleAdapter。功能强大的 Adapter,用于将 List 集合的多个对象包装成多个列表项。
- SimpleCursorAdapter。与 SimpleAdapter 相似,用于包装 Cursor 提供的数据。

使用 ArrayAdapter 为下拉列表加载数据,有两种方式:一是使用 Java 代码动态地定义下拉列表的数据源,一是使用 XML 文件预先定义数组资源描述文件。

下面分别介绍常用高级控件。

5.2.1 AutoCompleteTextView

AutoCompleteTextView(自动完成文本框)继承自 TextView 类,它也是一个文本框,但它多了一个功能:当用户输入一定的字符串后,它会弹出一个下拉菜单,供用户选择,用户选择某个菜单后,AutoCompleteTextView 会按用户选择自动填写该文本框。

【例 5.5】 自动完成文本框举例。

【解题思路】

使用 String[]数据源创建一个 ArrayAdapter,为下拉列表进行数据绑定。

本例使用 Java 代码动态地定义下拉列表的数据源。

实例化 ArrayAdapter 格式如下:

```
public Array Adapter (Context context, int textViewResourceId, T[] objects)
```

参数 context 为当前的上下文对象,通常使用 this。

参数 textViewResourceId 为一个包含 TextView 的布局 XML 文件的 ID,用于告诉系统用来填充数据的布局方式。

参数 objects 为给 ArrayAdapter 提供数据的数组,用于填充下拉列表。

【开发步骤和程序分析】

(1) 在 Eclipse 中创建一个 AutoCompleteTextViewExample 应用项目,包名为 com.application.autocompletetextview。

(2) 设计布局,在 res/layout 目录下的 main.xml 文件中,设置一个 AutoCompleteTextView 控件。

在该文件中编辑代码如下:

```
<?xml version = "1.0" encoding = "utf - 8"?>
< LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = "vertical"
    android:layout_width = "fill_parent"
   android:layout_height = "wrap_content">
    <! -- 设置一个 AutoCompleteTextView 控件 -- >
```

```
7   <AutoCompleteTextView android:id = "@ + id/auto_complete"
8                 android:layout_width = "fill_parent"
9                 android:layout_height = "wrap_content"/>
10 </LinearLayout>
```

① 第 2 行至第 10 行定义一个垂直线性布局。

② 第 7 行至第 9 行定义一个 AutoCompleteTextView 控件，第 7 行定义其资源 ID 为 auto_complete。

(3) 在 com.application.autocompletetextview 包下的 AutoCompleteTextViewActivity.java 文件中，创建一个适配器并将其实例化，将适配器添加到 AutoCompleteTextView 控件对象上。

在该文件中编辑代码如下：

```
package com.application.autocompletetextview;

import com.application.autocompletetextview.R;
import android.app.Activity;
import android.os.Bundle;
import android.widget.ArrayAdapter;
import android.widget.AutoCompleteTextView;

public class AutoCompleteTextViewActivity extends Activity {

        //重写 onCreate()方法
        @Override
        protected void onCreate(Bundle savedInstanceState) {
              super.onCreate(savedInstanceState);
              setContentView(R.layout.autocomplete);
              setTitle("AutoCompleteTextViewExample");
              //创建一个适配器并将其实例化
              ArrayAdapter<String> adapter  =  new ArrayAdapter<String>(
                      this, android.R.layout.simple_dropdown_item_1line, FRUITS);
              //获取这个控件对象 autotextView
              AutoCompleteTextView autotextView  =  (AutoCompleteTextView) findViewById(
                      R.id.auto_complete);
              //使用 setAdapter(adapter)设置适配器
              autotextView.setAdapter(adapter);
              //定义用户需要输入的字符数为 1
              autotextView.setThreshold(1);
         }

         //定义一个常量数组 FRUITS,作为适配器的资源数组
         static final String[] FRUITS  =  new String[] {
         "apple" ,"aubergine","banana", "bean","berries",
         "broccoli" ,"cabbage" ,"celery", "cherries","coconut"
```

```
33          };
34 }
```

① 第6行至第7行，分别引入 android. widget. ArrayAdapter 类和 android. widget. AutoCompleteTextView 类，在代码中要使用 ArrayAdapter 类和 AutoCompleteTextView 类的对象。

② 第12行至第27行，重写 onCreate()方法。

③ 第18行，创建一个适配器并将其实例化。其中，参数 this 为当前的上下文对象，参数 android. R. layout. simple_dropdown_item_1line 为 Android 自带的简单布局，参数 FRUITS 为给适配器提供的数组。

④ 第21行，获取这个控件对象 autotextView。

⑤ 第24行，使用 setAdapter(adapter)设置适配器。

⑥ 第26行，定义用户需要输入的字符数为1。

⑦ 第30行至第33行，定义一个常量数组 FRUITS，作为适配器的资源数组。

【运行结果】

在 Eclipse 中启动模拟器，然后运行项目，运行结果如图5.7所示。

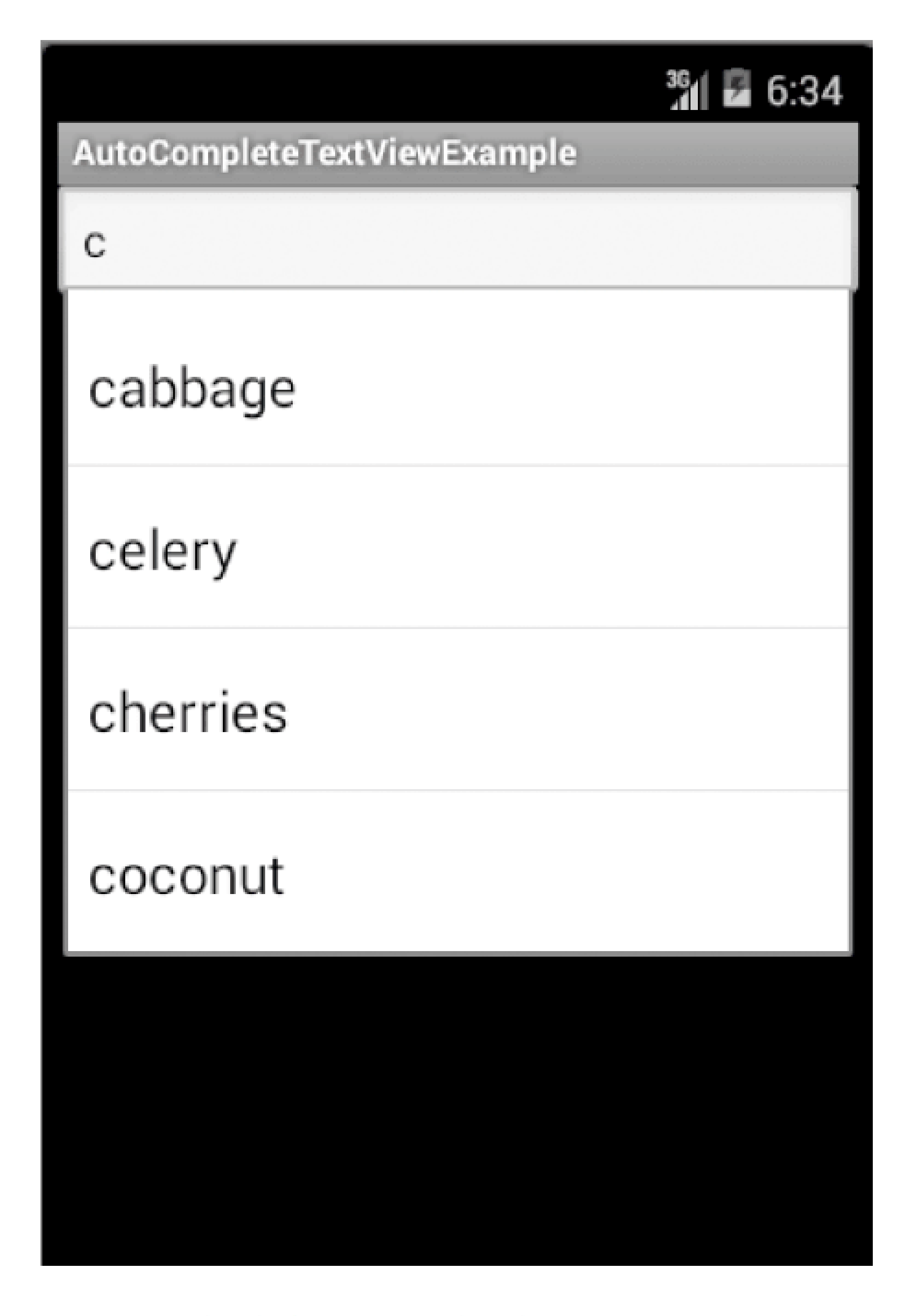


图5.7　自动完成文本框输入 c 时显示的界面

5.2.2　Spinner

Spinner 是一个垂直下拉列表框，位于 android. widget 包下，只有当用户单击这个控件时，才会下拉出选项列表供用户选择。

在下拉列表中的选项内容，需要绑定到数据源上，绑定数据需要用到适配器(Adapter)。

实现一个 Spinner 需要完成以下步骤：

(1) 为下拉列表项定义数据源；

(2) 实例化一个适配器；

(3) 为 Spinner 设置下拉列表下拉时的显示样式；

(4) 将适配器添加到 Spinner 上；

(5) 为 Spinner 添加监听器，设置各种事件的响应操作。

【例 5.6】 Spinner 举例。

【解题思路】

创建 Spinner，使用垂直下拉列表选择所在的城市。本例使用 XML 文件预先定义数组资源描述文件。

【开发步骤和程序分析】

(1) 在 Eclipse 中创建一个 SpinnerExample 应用项目，包名为 com.application.spinnerexample。

(2) 设计布局，在 res/layout 目录下的 main.xml 文件中，设置一个 TextView 控件、一个 Spinner 控件。

在该文件中编辑代码如下：

```
1  <?xml version = "1.0" encoding = "utf - 8"?>
2  < LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
3      android:id = "@ + id/widget28"
4      android:layout_width = "fill_parent"
5      android:layout_height = "fill_parent"
6      android:orientation = "vertical"
7      >
8      <! -- 设置一个 TextView 控件 -- >
9      < TextView
10         android:id = "@ + id/TextView_Show"
11         android:layout_width = "fill_parent"
12         android:layout_height = "wrap_content"
13         android:text = "选择所在城市"
14         android:textSize = "20sp"/>
15     <! -- 设置一个 Spinner 控件 -- >
16     < Spinner
17         android:id = "@ + id/spinner_City"
18         android:layout_width = "fill_parent"
19         android:layout_height = "wrap_content"/>
20 </LinearLayout >
```

① 第 2 行至第 20 行定义一个垂直线性布局。

② 第 9 行至第 14 行定义一个 TextView 控件，用于显示从 Spinner 下拉列表中选择的内容。

③ 第 16 行至第 19 行定义一个 Spinner 控件。

(3) 创建资源数组文件,在 res/values 目录下的 arrays. xml 文件中,编辑代码如下:

```
1  <?xml version = "1.0" encoding = "utf - 8"?>
2  <resources>
3      <string - array name = "citys">
4          <item>北京</item>
5          <item>上海</item>
6          <item>广州</item>
7          <item>成都</item>
8      </string - array>
9  </resources>
```

第 3 行定义资源数组名称为 citys。

(4) 在 src/com. application. spinnerexample 包下的 SpinnerExampleActivity. java 文件中,创建一个适配器并将其实例化,将适配器添加到 Spinner 控件对象上,为 Spinner 对象添加一个 OnItemSelectedListener()监听。

在该文件中编辑代码如下:

```
package com. application. spinnerexample;

import com. application. spinnerexample. R;
import android. app. Activity;
import android. os. Bundle;
import android. view. View;
import android. widget. AdapterView;
import android. widget. ArrayAdapter;
import android. widget. Spinner;
import android. widget. TextView;

public class SpinnerExampleActivity extends Activity {
    private TextView text;
    private Spinner spinner;

    //重写 onCreate()方法
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super. onCreate(savedInstanceState);
        setContentView(R. layout. main);
        text = (TextView)findViewById(R. id. TextView_Show);
        spinner = (Spinner)findViewById(R. id. spinner_City);
        //创建一个名为 adapter 的 ArrayAdapter 的实例
        ArrayAdapter<CharSequence> adapter  = ArrayAdapter. createFromResource
            (this, R. array. citys, android. R. layout. simple_spinner_item);
        //设置 adapter 将要绑定的 Spinner 的下拉列表显示样式
```

```
27          adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
28          //将 adapter 添加到 spinner 中
29          spinner.setAdapter(adapter);
30          //设置 Spinner 的属性
31          spinner.setPrompt("请选择所在的城市：");
32          spinner.setSelection(0, true);
33          //为 Spinner 添加一个 OnItemSelectedListener()监听
34          spinner.setOnItemSelectedListener(new Spinner.OnItemSelectedListener(){
35
36              @Override
37              public void onItemSelected(AdapterView<?> arg0, View arg1,int arg2, long arg3) {
38                  text.setText("你所在的城市是：" + arg0.getItemAtPosition(arg2).toString());
39                  arg0.setVisibility(View.VISIBLE);
40              }
41
42              @Override
43              public void onNothingSelected(AdapterView<?> arg0) {
44              }
45          });
46      }
47 }
```

① 第 6 行至第 10 行，分别引入 android.view.View、android.widget.AdapterView、android.widget.ArrayAdapter、android.widget.Spinner、android.widget.TextView 类，在代码中要使用 View、AdapterView、ArrayAdapter、Spinner 和 TextView 类的对象。

② 第 17 行至第 46 行，重写 onCreate()方法。

③ 第 24 行，创建一个名为 adapter 的 ArrayAdapter 的实例，ArrayAdapter 的数据来自 arrays.xml 资源数组文件。

④ 第 27 行，设置 adapter 将要绑定的 Spinner 的下拉列表显示样式。

⑤ 第 29 行，将 adapter 添加到 spinner 中。

⑥ 第 31 行至第 32 行，设置 Spinner 的一些属性，第 25 行设置该下拉列表的提示信息。

⑦ 第 34 行至第 45 行，为 Spinner 对象添加一个 OnItemSelectedListener()监听，其中重定义一些回调方法。

【运行结果】

在 Eclipse 中启动模拟器，然后运行项目 SpinnerExample，初始界面如图 5.8 所示；单击 Spinner 控件后，下拉出选项列表，如图 5.9 所示；



图 5.8　Spinner 下拉列表初始界面

在下拉列表中选择“上海”后，显示结果如图 5.10 所示。



图 5.9　单击下拉列表后下拉的列表选项



图 5.10　选择城市“上海”后显示的界面

5.2.3　Gallery

Gallery(画廊视图)是水平滚动显示图片资源的列表。

Gallery 和 Spinner 都是列表框，它们有共同的父类 AbsSpinner。它们的区别是：Gallery 是一个水平的列表选择框，Spinner 是一个垂直的列表选择框；Gallery 通过用户拖动来查看列表项，Spinner 通过用户选择来查看列表项。

Gallery 的属性与方法如表 5.2 所示。

表 5.2　Gallery 的属性与方法

属　　性	方　　法	说　　明
android:animationDuration	setAnimationDuration ()	设置动画过渡时间
android:gravity	setGravity()	设置在父控件中的对齐方式
android:unselectedAlpha	setUnselectedAlpha ()	设置选中的图片透明度
android:spacing	setSpacing ()	设置图片之间的空白大小

【例 5.7】 创建画廊视图，使用水平滚动列表显示多个图片。

【解题思路】

使用 BaseAdapter 创建一个 galleryadapter 对象并实例化。

【开发步骤和程序分析】

(1) 在 Eclipse 中创建一个 GalleryExample 应用项目，包名为 com. application.

galleryexample。

(2) 准备资源，将图片资源复制到 res/drawable-mdpi 中。图片资源名为 photo01.jpg～photo04.jpg。

(3) 设计布局，在 res/layout 目录下的 main.xml 文件中设置一个 Gallery 控件。

在该文件中编辑代码如下：

```
1  <?xml version = "1.0" encoding = "utf - 8"?>
2  <LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
3      android:orientation = "vertical"
4      android:layout_width = "fill_parent"
5      android:layout_height = "fill_parent"
6      android:gravity = " center_vertical"
7      android:id = "@ + id/llayout"
8      >
9      <! -- 设置一个 Gallery 控件 -->
10     <Gallery
11         android:id = "@ + id/gallery1"
12         android:spacing = "16dip"
13         android:unselectedAlpha = "1"
14         android:layout_width = "match_parent"
15         android:layout_height = "wrap_content" />
16 </LinearLayout>
```

① 第 2 行至第 16 行定义一个垂直线性布局。

② 第 10 行至第 15 行定义一个 Gallery 控件；第 13 行设置被选中项不透明度为 100%，即显示照片最清晰。

(4) 在 src/com.application.galleryexample 包下的 GalleryExampleActivity.java 文件中，创建一个适配器并将其实例化，将适配器添加到 Gallery 控件对象上，为 Gallery 对象添加一个 OnItemSelectedListener()监听。

在该文件中编辑代码如下：

```
package com.application.galleryexample;

import com.application.galleryexample.R;
import android.app.Activity;
import android.content.res.TypedArray;
import android.os.Bundle;
import android.view.View;
import android.view.ViewGroup;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.BaseAdapter;
import android.widget.Gallery;
import android.widget.ImageView;
```

```java
import android.widget.Toast;

public class GalleryExampleActivity extends Activity {
  //定义图片资源的 id 的数组 imageId
  private int[] imageId = new int[] { R.drawable.photo01, R.drawable.photo02,
           R.drawable.photo03, R.drawable.photo04 };

  @Override
   public void onCreate(Bundle savedInstanceState) {
       super.onCreate(savedInstanceState);
       setContentView(R.layout.main);
       //获取 Gallery 组件
       Gallery gallery = (Gallery) findViewById(R.id.gallery1);

       //创建 BaseAdapter 对象 galleryadapter 并实例化
       BaseAdapter galleryadapter = new BaseAdapter() {

           @Override
           public View getView(int position, View convertView, ViewGroup parent) {
               ImageView imageview;
               if (convertView == null) {
                   imageview = new ImageView(GalleryExampleActivity.this);
                   imageview.setScaleType(ImageView.ScaleType.FIT_XY);
                   imageview.setLayoutParams(new Gallery.LayoutParams(240, 180));
                   TypedArray typedArray = obtainStyledAttributes(R.styleable.Gallery);
                   imageview.setBackgroundResource(typedArray.getResourceId(
                           R.styleable.Gallery_android_galleryItemBackground,
                           0));
                   imageview.setPadding(5, 0, 5, 0);
               } else {
                   imageview = (ImageView) convertView;
               }
               imageview.setImageResource(imageId[position]);
               return imageview;
           }

           @Override
           public long getItemId(int position) {
               return position;
           }

           @Override
           public Object getItem(int position) {
               return position;
           }
```

```
59
60             @Override
61             public int getCount() {
62                 return imageId.length;
63             }
64         };
65         //将适配器与 Gallery 关联
66         gallery.setAdapter(galleryadapter);
67         gallery.setSelection(imageId.length / 2);
68
69         //为 gallery 对象添加一个 OnItemClickListener 监听
70         gallery.setOnItemClickListener(new OnItemClickListener() {
71
72             @Override
73             public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
74                 Toast.makeText(GalleryExampleActivity.this,
75                         "您单击了第" + String.valueOf(position) + "张图片",
76                         Toast.LENGTH_SHORT).show();
77             }
78         });
79     }
80 }
```

① 第 18 行，定义图片资源的 id 的数组 imageId。

② 第 26 行，获取 Gallery 组件。

③ 第 29 行至第 64 行，创建 BaseAdapter 对象 galleryadapter 并实例化。其中，第 33 行声明 ImageView 的对象，第 35 行实例化 ImageView 的对象，第 36 行设置缩放方式，第 42 行设置 ImageView 的内边距，第 46 行为 ImageView 设置要显示的图片，第 47 行返回 ImageView，第 50 行至第 53 行获取当前选项的 ID，第 55 行至第 58 行获取当前选项，第 60 行至第 63 行获取数量。

④ 第 66 行将适配器与 Gallery 关联，第 67 行让中间的图片选中。

⑤ 第 70 行至第 78 行，为 gallery 对象添加一个 OnItemClickListener 监听，其中重定义回调方法。

【运行结果】

在 Eclipse 中启动模拟器，然后运行项目 GalleryExample，画廊视图的初始界面如图 5.11 所示；单击画廊视图内的图片时显示的界面如图 5.12 所示。

图 5.11　画廊视图的初始界面



图 5.12 单击画廊视图内的图片时显示的界面

5.2.4 ListView

ListView(列表视图)以垂直的可滚动的列表方式显示一组列表项的视图。

实现一个 ListView 控件,有以下 4 个步骤:

(1) 准备 ListView 要显示的数据,使用一维或多维动态数组保存数据。

(2) 构建适配器。由于 ListView 的每一个 Item 的组成可能简单,也可能复杂,根据需要,可选择 ArrayAdapter、SimpleAdapter 或 BaseAdapter 来为 ListView 绑定数据。

(3) 使用 setAdapter(),把适配器添加到 ListView,并显示出来。

(4) 为 ListView 添加监听器,设置各种事件(如单击、滚动、单击长按等)的响应操作。

ListView 常用的监听包括:

(1) 单击监听,添加单击监听使用 ListView. setOnItemClickListener();

(2) 滚动监听,添加滚动监听使用 ListView. setOnItemSelectedListener();

(3) 长按监听,添加长按监听使用 setOnCreateContextMenuListener()。

1. 使用 ArrayAdapter 适配器

使用 ArrayAdapter 适配器为 ListView 绑定数据,可以创建每条目显示一行字符串 ListView 控件,实现方法已在例 5.5 和例 5.6 中做了介绍。

2. 使用 SimpleAdapter 适配器

SimpleAdapter 适配器的扩展性最好,可以定义许多形式的布局,例如可放上 Imageview(图片)、Button(按钮)、CheckBox(复选框)等。使用 SimpleAdapter 适配器构造

数据，一般是用数组列表 ArrayList，而 ArrayList 一般通过 HashMap 构成，HashMap 是一组键-值对的集合。

【例 5.8】 创建列表视图，使用垂直列表列出一些公司的网址信息，单击某一项目时，在标题栏显示其网址信息。

【解题思路】

使用 SimpleAdapter 适配器构造数据需要用到数组列表 ArrayList，其中的 HashMap 对象对应于 ListView 每一 Item（条目）。在本例中，ListView 中每一 Item 包含一个 Imageview 控件和两个分上下行的 TextView 控件，该布局可用 res/layout 目录下的 XML 文件来定义。

本例要求，单击 ListView 一个 Item 项，在标题栏显示其网址信息，需要为 ListView 对象添加监听，重写回调方法。

【开发步骤和程序分析】

(1) 在 Eclipse 中创建一个 ListViewSimpleExample 应用项目，包名为 com.application.listviewsimpleexample。

(2) 准备资源，将图片资源复制到 res/drawable-mdpi 中，图片资源名为 acompany.png。

(3) 设计布局，在 res/layout 目录下的 main.xml 文件中设置一个 ListView 控件。

在该文件中编辑代码如下：

```
1  <?xml version = "1.0" encoding = "utf - 8"?>
2  <LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
3      android:id = "@ + id/LinearLayout01"
4      android:layout_width = "fill_parent"
5      android:layout_height = "fill_parent"
6      >
7       <!-- 设置一个 ListView 控件 -->
8        <ListView
9          android:id = "@ + id/ListView01"
10         android:layout_width = "fill_parent"
11         android:layout_height = "wrap_content"
12         />
13 </LinearLayout>
```

① 第 2 行至第 13 行定义一个线性布局。

② 第 8 行至第 12 行定义一个 ListView 控件，其 ID 名为 ListView01。

(4) 设计 ListView 的条目布局，在 res/values 目录下的 listitem.xml 文件中，编辑代码如下：

```
<?xml version = "1.0" encoding = "utf - 8"?>
<!-- 定义 ListView 的条目布局为一个水平线性布局 -->
<LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = "horizontal"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent">
    <!-- 设置一个 ImageView 控件 -->
```

```
8       <ImageView android:id="@+id/img"
9           android:layout_width="45dip"
10          android:layout_height="45dip"
11          android:layout_margin="10px"/>
12      <!-- 定义一个嵌套的垂直线性布局 -->
13      <LinearLayout android:orientation="vertical"
14          android:layout_width="wrap_content"
15          android:layout_height="wrap_content">
16          <!-- 设置第一行的 TextView 控件 -->
17          <TextView android:id="@+id/title"
18              android:layout_width="wrap_content"
19              android:layout_height="wrap_content"
20              android:textColor="#FFFFFFFF"
21              android:textSize="22px" />
22          <!-- 设置第二行的 TextView 控件 -->
23          <TextView android:id="@+id/info"
24              android:layout_width="wrap_content"
25              android:layout_height="wrap_content"
26              android:textColor="#FFFFFFFF"
27              android:textSize="13px" />
28      </LinearLayout>
29  </LinearLayout>
```

① 第 2 行至第 29 行定义 ListView 的条目布局为一个水平线性布局。

② 第 8 行至第 11 行定义一个 ImageView 控件，其 ID 名为 img。为了使 ListView 每一 Item 的高度一致，显示的图片大小也相应一致，设定 ImageView 的高度和宽度值。

③ 第 13 行至第 28 行定义一个嵌套的垂直线性布局，用于定义两行文本控件。

④ 第 17 行至第 21 行定义第一行的 TextView 控件，其 ID 名为 title，设置了文本的大小和颜色。

⑤ 第 23 行至第 27 行定义第二行的 TextView 控件，其 ID 名为 info，设置了文本的大小和颜色。

(5) 在 src/ListViewSimpleExampl 包下的 ListViewSimpleExampleActivity.java 文件中，创建一个适配器并将其实例化，将适配器添加到 ListView 控件对象上，为 ListView 对象添加监听器。

```
package com.application.listviewsimpleexample;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import com.application.listviewsimpleexample.R;
import android.app.Activity;
import android.os.Bundle;
```

```
import android.widget.ListView;
import android.widget.SimpleAdapter;
import android.view.View;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;

public class ListViewSimpleExampleActivity extends Activity {

        //重写 onCreate()方法
        @Override
        public void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.main);
            //ListView 类的对象 list 获取 ID 为 ListView01 的资源
            ListView list = (ListView) findViewById(R.id.ListView01);

            //创建 SimpleAdapter 类的对象 listItemAdapter,
            //并生成适配器的 Item 和动态数组对应的元素
            SimpleAdapter listItemAdapter = new SimpleAdapter(
                    this,
                    getData(),
                    R.layout.listitem,
                    new String[]{"img","title","info"},
                    new int[]{R.id.img,R.id.title,R.id.info});

            //将适配器添加到 ListView 对象 list
            list.setAdapter(listItemAdapter);

            //为 list 添加监听器
            list.setOnItemClickListener(new OnItemClickListener() {
                @Override
                public void onItemClick(AdapterView<?> arg0, View arg1, int arg2, long arg3) {
                  Map<String, Object> clkmap = (Map<String, Object>)
                        arg0.getItemAtPosition(arg2);
                  setTitle(clkmap.get("title").toString() + "的网址是: " +
                        clkmap.get("info").toString());
                }
            });
        }

        //生成多维动态数组,并加入数据
        private List<Map<String, Object>> getData() {
            //创建 ArrayList 类的对象 listItem
            ArrayList<Map<String, Object>> listitem = new ArrayList
                    <Map<String, Object>>();
```

```
55            //创建 HashMap 类的对象 map,map 内有三组键 - 值对,分别是 img、title、info
56            Map<String, Object> map = new HashMap<String, Object>();
57
58            map.put("img", R.drawable.acompany);
59            map.put("title", "A 公司");
60            map.put("info", "http://www.aaa.com/");
61            //将 map 对象加入到 listItem 列表中
62            listitem.add(map);
63
64            map = new HashMap<String, Object>();
65            map.put("img", R.drawable.acompany);
66            map.put("title", "B 公司");
67            map.put("info", "http://www. bbb.com /");
68            listitem.add(map);
69
70            map = new HashMap<String, Object>();
71            map.put("img", R.drawable.acompany);
72            map.put("title", "C 公司");
73            map.put("info", "http://www. ccc.com /");
74            listitem.add(map);
75
76            //返回赋值的 listitem 对象
77            return listitem;
78        }
79
80 }
```

① 第 3 行至第 6 行,分别引入 Java 类 ArrayList、HashMap、List 和 Map。

② 第 8 行至第 14 行,分别引入 Android 中的相关类。

③ 第 19 行至第 48 行重写 onCreate()方法。其中,第 24 行 ListView 类的对象 list 获取 ID 为 ListView01 的资源,第 28 行至第 33 行创建 SimpleAdapter 类的对象 listItemAdapter,并生成适配器的 Item 和动态数组对应的元素,第 30 行定义的 getData()方法返回一个 ArrayList 列表,第 39 行至第 47 行添加单击监听。

④ 第 51 行至第 78 行生成多维动态数组,并加入数据。其中,第 56 行创建 ArrayList 类的对象 listItem,创建时使用:

```
ArrayList<Map<String, Object>> listitem = new ArrayList<Map<String, Object>>();
```

这里的 String 指定键名的类型,Object 指定键值的类型,第 55 行创建 HashMap 类的对象 map:

```
Map<String, Object> map = new HashMap<String, Object>();
```

map 内有三组键-值对,分别是 img、title、info,img 的键值是图片资源,第 58 行至第 60 行分别为 map 的键名传入相应的键值内容,第 62 行将 map 对象加入到 listItem 列表中,第 77

行返回赋值的 listitem 对象。

【运行结果】

在 Eclipse 中启动模拟器,然后运行项目 ListViewSimpleExample,列表视图的初始界面如图 5.13 所示;单击列表视图第一项时显示的界面如图 5.14 所示。

3. 使用 BaseAdapter 适配器

BaseAdapter 适配器直接继承接口类 Adapter,在 BaseAdapter 中,需要在 BaseAdapter 类定义中重写 getCoumt()、getItem()、getItemID()、getView()等方法。

【例 5.9】 在例 5.8 的列表视图中,每个项目增加一个按钮控件,单击某一项目的按钮时,在标题栏显示其网址信息。

【解题思路】

使用动态生成显示布局的方式,需要在 BaseAdapter 类定义中重写 getView()方法,并为每一条目中的按钮添加按钮单击监听方法 OnItemClickListener()。



图 5.13 列表视图的初始界面



图 5.14 单击列表视图第一项时显示的界面

【开发步骤和程序分析】

(1) 在 Eclipse 中创建一个 ListViewBaseExample 应用项目,包名为 com.application.listviewbaseexample。

(2) 准备图片资源,将图片资源复制到 res/drawable-mdpi 中,图片资源名为 acompany.png。

(3) 准备字符串资源,在 res/values 目录下的 string.xml 文件中,编辑代码如下:

```
<?xml version = "1.0" encoding = "utf - 8"?>
```

```
2 <resources>
3     <string name = "hello">Hello World, ListViewBaseExample!</string>
4     <string name = "app_name">ListViewBaseExample</string>
5     <string name = "aname">A 公司</string>
6     <string name = "bname">B 公司</string>
7     <string name = "cname">C 公司</string>
8     <string name = "aurl">http://www.aaa.com/</string>
9     <string name = "burl">http://www.bbb.com/</string>
10    <string name = "curl">http://www.ccc.com/</string>
11 </resources>
```

第 5 行至第 10 行声明了 3 对字符串信息，分别是大学的校名和网址。

(4) 创建颜色资源，创建并编写 res/values 目录下的颜色描述文件 colors.xml 文件，代码如下：

```
<?xml version = "1.0" encoding = "utf - 8"?>
<resources>
    <color name = "white">#FFFFFFFF</color>
    <color name = "red">#FFFD8D8D</color>
    <color name = "blue">#FF8D9DFD</color>
</resources>
```

(5) 创建数组资源，创建并编写 res/values 目录下的描述文件 arrays.xml 文件，代码如下：

```
1 <?xml version = "1.0" encoding = "utf - 8"?>
2 <resources>
3     <string - array name = "images">
4         <item>acompany</item>
5         <item>acompany</item>
6         <item>acompany</item>
7     </string - array>
8     <string - array name = "titles">
9         <item>aname</item>
10        <item>bname</item>
11        <item>cname</item>
12    </string - array>
13    <string - array name = "infos">
14        <item>aurl</item>
15       <item>burl</item>
16       <item>curl</item>
17    </string - array>
18 </resources>
```

① 第 3 行至第 7 行定义一组图片资源的 ID 名数组 images。

② 第 8 行至第 12 行定义一组内容，网站名称字符串资源的 ID 名数组 titles。

③ 第 13 行至第 17 行定义一组内容，网站地址字符串资源的 ID 名数组 infos。

(6) 设计布局,在 res/layout 目录下的 main. xml 文件中,设置一个 ListView 控件,其 ID 名为 ListView02。

在该文件中编辑代码如下:

```
1  <?xml version = "1.0" encoding = "utf - 8"?>
2  <LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
3      android:id = "@ + id/LinearLayout01"
4      android:layout_width = "fill_parent"
5      android:layout_height = "fill_parent"
6      >
7       <!-- 设置一个 ListView 控件,其 ID 名为 ListView02 -->
8       <ListView
9         android:id = "@ + id/ListView02"
10        android:layout_width = "fill_parent"
11        android:layout_height = "wrap_content"
12        />
13 </LinearLayout>
```

① 第 2 行至第 13 行定义一个线性布局。

② 第 8 行至第 12 行定义一个 ListView 控件,其 ID 名为 ListView02。

(7) 在 src/com. application. listviewbaseexample 包下的 ListViewBaseExampleActivity. java 文件中,创建适配器 BaseAdapter 类的对象 baseadapter,动态生成每个下拉项对应的 View。每个下拉项 View 的 LinearLayout 中,由一个 ImageView、一个内嵌的 LinearLayout(其中包含两个 TextView),以及另一个内嵌的 LinearLayout(其中包含一个按钮并为按钮对象添加监听器)构成,将适配器添加到 ListView 控件对象上。

在该文件中编辑代码如下:

```
package com.application.listviewbaseexample;

import com.application.listviewbaseexample.R;
import android.app.Activity;
import android.os.Bundle;
import android.view.Gravity;
import android.view.View;
import android.view.ViewGroup;
import android.view.ViewGroup.LayoutParams;
import android.widget.AdapterView;
import android.widget.BaseAdapter;
import android.widget.Button;
import android.widget.Gallery;
import android.widget.ImageView;
import android.widget.LinearLayout;
import android.widget.ListView;
import android.widget.TextView;
```

```java
import android.widget.AdapterView.OnItemSelectedListener;

public class ListViewBaseExampleActivity extends Activity {

    //重写 onCreate()方法
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        //初始化 ListView
        ListView lv = (ListView)this.findViewById(R.id.ListView02);

        //创建 BaseAdapter 类的对象 baseadapter
        BaseAdapter baseadapter = new BaseAdapter(){

            //定义所有图片资源名的数组、所有标题资源名的数组、所有网址资源名的数组
            String[] images =  getResources().getStringArray(R.array.images);
            String[] titles = getResources().getStringArray(R.array.titles);
            String[] infos = getResources().getStringArray(R.array.infos);
            public int getCount() {return images.length;}
            public Object getItem(int arg0) { return null; }
            public long getItemId(int arg0) { return arg0; }

            //重写 getView()方法,动态生成每个下拉项对应的 View
            public View getView(int arg0, View arg1, ViewGroup arg2) {

                //创建一个水平方向的 LinearLayout 布局 ll0
                LinearLayout ll0 = new LinearLayout(ListViewBaseExampleActivity.this);
                ll0.setOrientation(LinearLayout.HORIZONTAL);

                //创建一个 ImageView 控件对象 img
                ImageView img = new ImageView(ListViewBaseExampleActivity.this);
                img.setImageDrawable(getResources().getDrawable(
                    getResources().getIdentifier(images[arg0],"drawable",getPackageName())));
                img.setLayoutParams(new Gallery.LayoutParams(45,45));
                img.setPadding(5, 5, 5, 5);
                ll0.addView(img);

                //创建一个内嵌的竖直方向的 LinearLayout 布局 ll1
                LinearLayout ll1 = new LinearLayout(ListViewBaseExampleActivity.this);
                ll1.setOrientation(LinearLayout.VERTICAL);
                ll1.setLayoutParams(new LinearLayout.LayoutParams(
                    LayoutParams.WRAP_CONTENT, LayoutParams.WRAP_CONTENT));

                //分别创建两个 TextView 控件对象并添加到布局 ll1 中
```

```
                TextView tit = new TextView(ListViewBaseExampleActivity.this);
                tit.setText(getResources().getText(getResources().
                    getIdentifier(titles[arg0],"string",getPackageName())));
                tit.setLayoutParams(new LayoutParams(LinearLayout.LayoutParams.
                    WRAP_CONTENT,LinearLayout.LayoutParams.WRAP_CONTENT));
                tit.setTextSize(18);
                tit.setTextColor(getResources().getColor(R.color.white));
                ll1.addView(tit);

                TextView inf = new TextView(ListViewBaseExampleActivity.this);
                TextView inf = new TextView(ListViewBaseExampleActivity.this);
                inf.setText(getResources().getText(
                        getResources().getIdentifier(infos[arg0],"string",getPackageName())));
                inf.setLayoutParams(newLayoutParams(LinearLayout.LayoutParams.
                        WRAP_CONTENT, LinearLayout.LayoutParams.WRAP_CONTENT));
                inf.setTextSize(13);
                inf.setTextColor(getResources().getColor(R.color.white));
                ll1.addView(inf);
                //将 ll1 添加到 ll0 中
                ll0.addView(ll1);

                //创建一个新的水平方向的 LinearLayout 布局 ll2
                LinearLayout ll2 = new LinearLayout(ListViewBaseExampleActivity.this);
                ll2.setOrientation(LinearLayout.HORIZONTAL);
                ll2.setLayoutParams(new LinearLayout.LayoutParams
                    (LayoutParams.FILL_PARENT, LayoutParams.WRAP_CONTENT));
                ll2.setPadding(10, 0, 10, 0);
                ll2.setGravity(Gravity.RIGHT);

                //在 ll2 内,创建一个按钮对象 btn
                Button btn = new Button(ListViewBaseExampleActivity.this);
                btn.setLayoutParams(new LinearLayout.LayoutParams(
                    50, LayoutParams.WRAP_CONTENT));
                btn.setText("显示网址");
                btn.setId(arg0);
                //为 btn 添加 OnClickListener()监听
                btn.setOnClickListener(new View.OnClickListener() {

                    @Override
                    public void onClick(View v) {
                        StringBuilder cb = new StringBuilder();
                        cb.append("单击第" + Integer.toString(v.getId() + 1) + "项,网址为: ");
                        cb.append(getResources().getText(
    getResources().getIdentifier(infos[v.getId()],"string",getPackageName())));
                        String clkstr = cb.toString();
```

```
108                             setTitle(clkstr);
109                         }
110                     });
111                     ll2.addView(btn);
112                     //将 ll2 添加到 ll0 中
113                     ll0.addView(ll2);
114                     return ll0;
115                 }
116         };
117 
118          //为 ListView 对象 lv 添加适配器 baseadapte
119         lv.setAdapter(baseadapter);
120 
121         //为 lv 对象添加滚动监听器
122         lv.setOnItemSelectedListener(
123             new OnItemSelectedListener(){
124                 public void onItemSelected(AdapterView<?> arg0, View arg1, int arg2, long arg3) {
125 
126                     LinearLayout ll0 = (LinearLayout)arg1;
127                     LinearLayout ll1 = (LinearLayout)ll0.getChildAt(1);
128 
129                     StringBuilder sb = new StringBuilder();
130                     TextView tv1 = (TextView)ll1.getChildAt(0);   /
131                     sb.append(tv1.getText() + "的网址为: ");
132                     TextView tv2 = (TextView)ll1.getChildAt(1);
133                     sb.append(tv2.getText());
134                     String stemp = sb.toString();
135                     setTitle(stemp);
136                 }
137                 public void onNothingSelected(AdapterView<?> arg0){}
138             }
139         );
140     }
141 }
```

① 第 4 行至第 18 行，分别引入 Android 中的相关类。

② 第 28 行初始化 ListView。

③ 第 31 行至第 116 行创建 BaseAdapter 类的对象 baseadapter，重写 baseadapter 中的 getCoumt()、getItem()、getItemID()、getView()等方法，添加按钮监听 OnItemClickListener()。

④ 第 34 行至第 36 行，定义所有图片资源名(acompany、acompany、acompany)的数组、所有标题资源名(aname、bname、cname)的数组、所有网址资源名(aurl、burl、curl)的数组。

⑤ 第 37 行至第 39 行，分别重写 getCoumt()、getItem()、getItemID()方法，第 31 行返回数组元素个数。

⑥ 第 42 行至第 116 行，重写 getView()方法，动态生成每个下拉项对应的 View。每个

下拉项 View 由包含一个 ImageView 的 LinearLayout 和两个内嵌的 LinearLayout 构成(其中一个包含两个 TextView,另一个包含一个按钮)。

⑦ 第 45 行至第 46 行,创建一个水平方向的 LinearLayout 布局 ll0。

⑧ 第 49 行至第 54 行,创建一个 ImageView 控件对象 img。其中,第 50 行至第 51 行使用了 4 个方法为 img 获取图片资源,第 54 行将 img 对象添加到名为 ll0 的 LinearLayout 中。

⑨ 第 58 行至第 59 行,创建一个内嵌的竖直方向的 LinearLayout 布局 ll1。

⑩ 第 63 行至第 70 行,第 72 行至第 80 行,分别创建两个 TextView 控件对象并添加到布局 ll1 中。

⑪ 第 82 行将 ll1 添加到 ll0 中。

⑫ 第 85 行至第 90 行,创建一个新的水平方向的 LinearLayout 布局 ll2。

⑬ 第 93 行至第 96 行,在 ll2 内创建一个按钮对象 btn;第 97 行设置按钮的 ID。

⑭ 第 99 行至第 110 行,为 btn 添加 OnClickListener()监听;第 111 行将 btn 添加到 ll2 中;第 113 行将 ll2 添加到 ll0 中;第 114 行返回 110 布局对象,完成 BaseAdapter 类的对象 baseadapter 的创建。

⑮ 第 119 行为 ListView 对象 lv 添加适配器 baseadapte。

⑯ 第 122 行至第 139 行,为 lv 对象添加滚动监听器。

【运行结果】

在 Eclipse 中启动模拟器,然后运行项目 ListViewBaseExample,列表视图的初始界面如图 5.15 所示;单击第 2 项条目中的按钮显示的界面如图 5.16 所示。



图 5.15　列表视图的初始界面



图 5.16　单击第 2 项条目中的按钮显示的界面

5.2.5 GridView

GridView(网格视图)是一种以二维表格形式显示控件的视图,所显示的控件来自 ListAdapter 适配器。

GridView 和 ListView 有共同的父类:AbsListView。它们都是列表项,其区别为 ListView 只显示一列,而 GridView 可以显示多列。

GridView 的属性与方法如表 5.3 所示。

表 5.3 GridView 的属性与方法

属　　性	方　　法	说　　明
android:columnWidth	setColumnWidth()	设置列的宽度
android:gravity	setGravity()	设置对齐方式
android:numColumns	setNumColumns ()	设置各个元素之间的水平距离
android:horizontalSpacing	setHorizontalSpacing ()	设置列数
android:verticalSpacing	setVerticalSpacing()	设置各个元素之间的竖直距离

【例 5.10】 创建网格视图,按行、列分布的方式显示多个图片的内容。

【解题思路】

定义继承 BaseAdapter 类的 PictureAdapter 适配器,再提供给 GridView 使用。

为使用 res/layout 目录下的 XML 布局文件,Android 提供了一个专门的类 LayoutInflater,它用来找 res/layout 目录下的布局文件并实例化。

【开发步骤和程序分析】

(1) 在 Eclipse 中创建一个 GridViewExample 应用项目,包名为 com.application.gridviewexample。

(2) 准备资源,将图片资源复制到 res/drawable-mdpi 中,图片资源名为 grid_view_01.jpg～grid_view_12.jpg。

(3) 设计布局,在 res/layout 目录下的 main.xml 文件中,设置一个 GridView 控件,其 ID 名为 gridview。

在该文件中编辑代码如下:

```
<?xml version = "1.0" encoding = "utf - 8"?>
<! -- 设置一个 GridView 控件,其 ID 名为 gridview -->
<GridView xmlns:android = "http://schemas.android.com/apk/res/android"
    android:id = "@ + id/gridview"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:numColumns = "auto_fit"
    android:verticalSpacing = "10dp"
    android:horizontalSpacing = "10dp"
    android:columnWidth = "90dp"
    android:stretchMode = "columnWidth"
    android:gravity = "center"
```

```
13 />
```

① 第 4 行声明这个 GridView 控件 ID 名为 gridview。

② 第 7 行设置该 GridView 的列数，auto_fit 表示列数设置为自动。

③ 第 11 行设置该 GridView 的缩放模式为按列宽度缩放。

(4) 设计 GridView 的单元格布局，在 res/values 目录下的 pic_item.xml 文件中，编辑代码如下：

```
1  <?xml version = "1.0" encoding = "utf - 8"?>
2  < LinearLayout
3        xmlns:android = "http://schemas.android.com/apk/res/android"
4        android:id = "@ + id/root"
5        android:orientation = "vertical"
6        android:layout_width = "wrap_content"
7        android:layout_height = "wrap_content"
8        android:layout_marginTop = "5dp"
9        >
10       < ImageView
11           android:id = "@ + id/image"
12           android:layout_width = "85dp"
13           android:layout_height = "85dp"
14           android:layout_gravity = "center"
15           android:scaleType = "centerCrop"
16           android:padding = "4dp"
17           />
18       < TextView
19           android:id = "@ + id/title"
20           android:layout_width = "wrap_content"
21           android:layout_height = "wrap_content"
22           android:layout_gravity = "center"
23           />
24 </LinearLayout >
```

① 第 2 行至第 24 行定义 GridView 的单元格布局为一个垂直线性布局。

② 第 10 行至第 17 行定义一个 ImageView 控件，用于显示单元格中的图片。

③ 第 18 行至第 23 行定义一个 TextView 控件，用于显示单元格的图片标题。

(5) 在 src/com.application.gridviewexample 包下的 GridViewExampleActivity.java 文件中，声明 GridViewExampleActivity 类继承自 Activity 类，为 GridView 对象添加监听器，定义一个继承 BaseAdapter 类的适配器 PictureAdapter，定义 ViewHolder 类，定义 Picture 类。

在该文件中编辑代码如下：

```
package com.application.gridviewexample;

import java.util.ArrayList;
```

```
import java.util.List;
import com.application.gridviewexample.R;
import android.app.Activity;
import android.content.Context;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.AdapterView;
import android.widget.BaseAdapter;
import android.widget.GridView;
import android.widget.ImageView;
import android.widget.TextView;
import android.widget.AdapterView.OnItemClickListener;

//定义 GridViewExampleActivity 类继承自 Activity 类
public class GridViewExampleActivity extends Activity {
    private GridView gridView;
    //定义图片 ID 数组
    private int[] imgs = new int[] {
            R.drawable.grid_view_01, R.drawable.grid_view_02, R.drawable.grid_view_03,
            R.drawable.grid_view_04, R.drawable.grid_view_05, R.drawable.grid_view_06,
            R.drawable.grid_view_07, R.drawable.grid_view_08,R.drawable.grid_view_09,
            R.drawable.grid_view_10, R.drawable.grid_view_11,R.drawable.grid_view_12
    };
    //定义图片编号数组
    private String[] tits = new String[] {
            "photo01", "photo02", "photo03",
            "photo04", "photo05", "photo06",
            "photo07", "photo08", "photo09",
            "photo10", "photo11", "photo12"
    };

   //重写 onCreate()方法
   @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        gridView  = (GridView) findViewById(R.id.gridview);
        PictureAdapter adapter = new PictureAdapter(tits, imgs, this);
        //为 GridView 对象 gridView 添加 OnItemClickListener 监听器
        gridView.setAdapter(adapter);
        gridView.setOnItemClickListener(new OnItemClickListener() {
            //重写 OnItemClick()方法
            public void onItemClick(AdapterView<?> parent, View v, int position, long id) {
```

```
                    if (position < 9) {
                        setTitle( "单击的图片是: photo0" + (position + 1));
                    } else  {
                        setTitle( "单击的图片是: photo" + (position + 1));
                    };
                }
            });
    }
}

//定义一个继承 BaseAdapter 类的适配器 PictureAdapter
class PictureAdapter extends BaseAdapter{
    private LayoutInflater inflater;
    private List<Picture> pictures;
    //定义类 PictureAdapter 的构造方法
    public PictureAdapter(String[] titles, int[] images, Context context) {
        super();
        //创建一个 ArrayList 对象 pictures
        pictures = new ArrayList<Picture>();
        inflater = LayoutInflater.from(context);
        //将指定的 Picture 类型元素添加到数组列表 pictures 中
        for (int i = 0; i < images.length; i++)
        {
            Picture pic = new Picture(titles[i], images[i]);
            pictures.add(pic);
        }
    }

    @Override
    public int getCount() {
        if (null != pictures) {
            return pictures.size();
        } else {
            return 0;
        }
    }
    @Override
    public Object getItem(int position) {
        return pictures.get(position);
    }
    @Override
    public long getItemId(int position) { return position; }
    //PictureAdapter 类实例化
    @Override
    public View getView(int position, View convertView, ViewGroup parent)
```

```
94        {
95            ViewHolder viewHolder;
96            if (convertView == null)
97            {
98                convertView = inflater.inflate(R.layout.pic_item, null);
99                viewHolder = new ViewHolder();
100               viewHolder.title = (TextView) convertView.findViewById(R.id.title);
101               viewHolder.image = (ImageView) convertView.findViewById(R.id.image);
102               convertView.setTag(viewHolder);
103           } else
104           {
105               viewHolder = (ViewHolder) convertView.getTag();
106           }
107           viewHolder.title.setText(pictures.get(position).getTitle());
108           viewHolder.image.setImageResource(pictures.get(position).getImageId());
109           return convertView;
110       }
111 }
112
113 //定义 ViewHolder 类
114 class ViewHolder {
115       public TextView title;
116       public ImageView image;
117 }
118
119 //定义 Picture 类
120 class Picture {
121        private String title;
122        private int imageId;
123        public Picture() {
124            super();
125        }
126        public Picture(String title, int imageId) {
127            super();
128            this.title = title;
129            this.imageId = imageId;
130        }
131        public String getTitle() { return title; }
132        public void setTitle(String title) { this.title = title; }
133        public int getImageId() { return imageId; }
134        public void setImageId(int imageId) { this.imageId = imageId; }
135 }
```

① 第 3 行至第 17 行，分别引入 Java 和 Android 中的相关类。

② 第 20 行至第 57 行，定义 GridViewExampleActivity 类。其中，第 23 行至第 28 行定

义图片 ID 数组,第 30 行至第 35 行定义图片编号数组,第 38 行至第 56 行重写 onCreate()方法,第 43 行通过一个自定义适配器 PictureAdapter 创建适配器对象 adapter,第 45 行至第 55 行为 GridView 对象 gridView 添加 OnItemClickListener 监听器,并重写 OnItemClick()方法。

③ 第 60 行至第 111 行定义一个继承 BaseAdapter 类的适配器 PictureAdapter。其中,第 64 行至第 75 行定义类 PictureAdapter 的构造方法,第 66 行创建一个 ArrayList 对象 pictures,第 68 行获取 pictures 中每个 Item 的布局,第 70 行至第 74 行将指定的 Picture 类型元素添加到数组列表 pictures 中。

④ 第 92 行至第 110 行为 PictureAdapter 类实例化。其中,第 98 行通过 res/values 目录下的 pic_item.xml 文件获得每个网格的显示效果,第 99 行添加一个 ViewHolder 类的对象 viewHolder,第 102 行 convertView 中的 setTag(viewHolder)表示给 convertView 添加一个额外的数据,以后可用 getTag(viewHolder)将这个数据取出,第 107 行为 viewHolder 对象赋予相应的图片编号字符串并显示,第 108 行为 viewHolder 对象赋予相应的图片并显示。

⑤ 第 114 行至第 117 行,定义 ViewHolder 类。

⑥ 第 120 行至第 135 行,定义 Picture 类。

【运行结果】

在 Eclipse 中启动模拟器,然后运行项目 GridViewExample,运行结果如图 5.17 所示;单击了某个网格图片后,在标题栏显示相应的图片编号字符串,如图 5.18 所示。

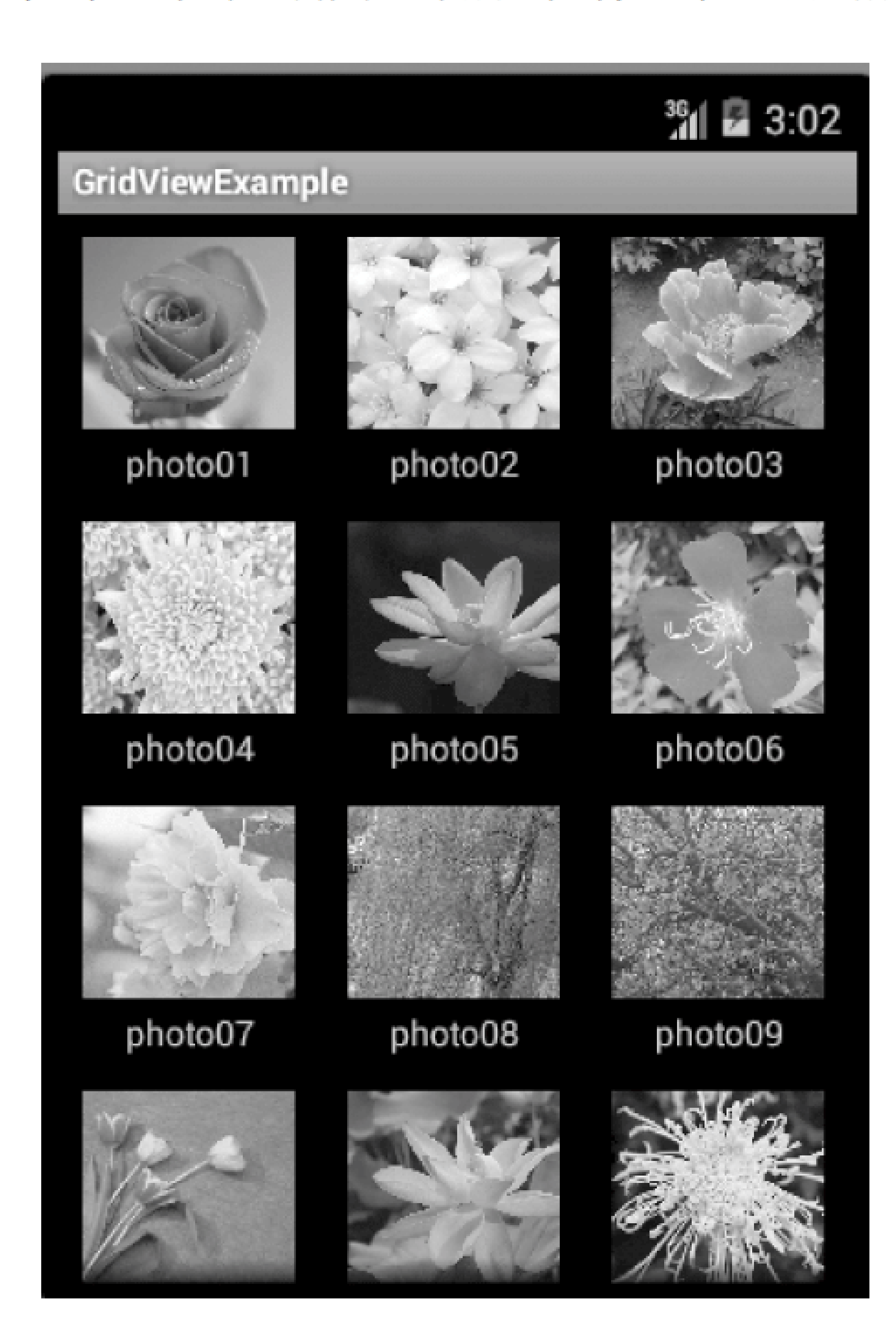


图 5.17 初始运行时网格视图的界面

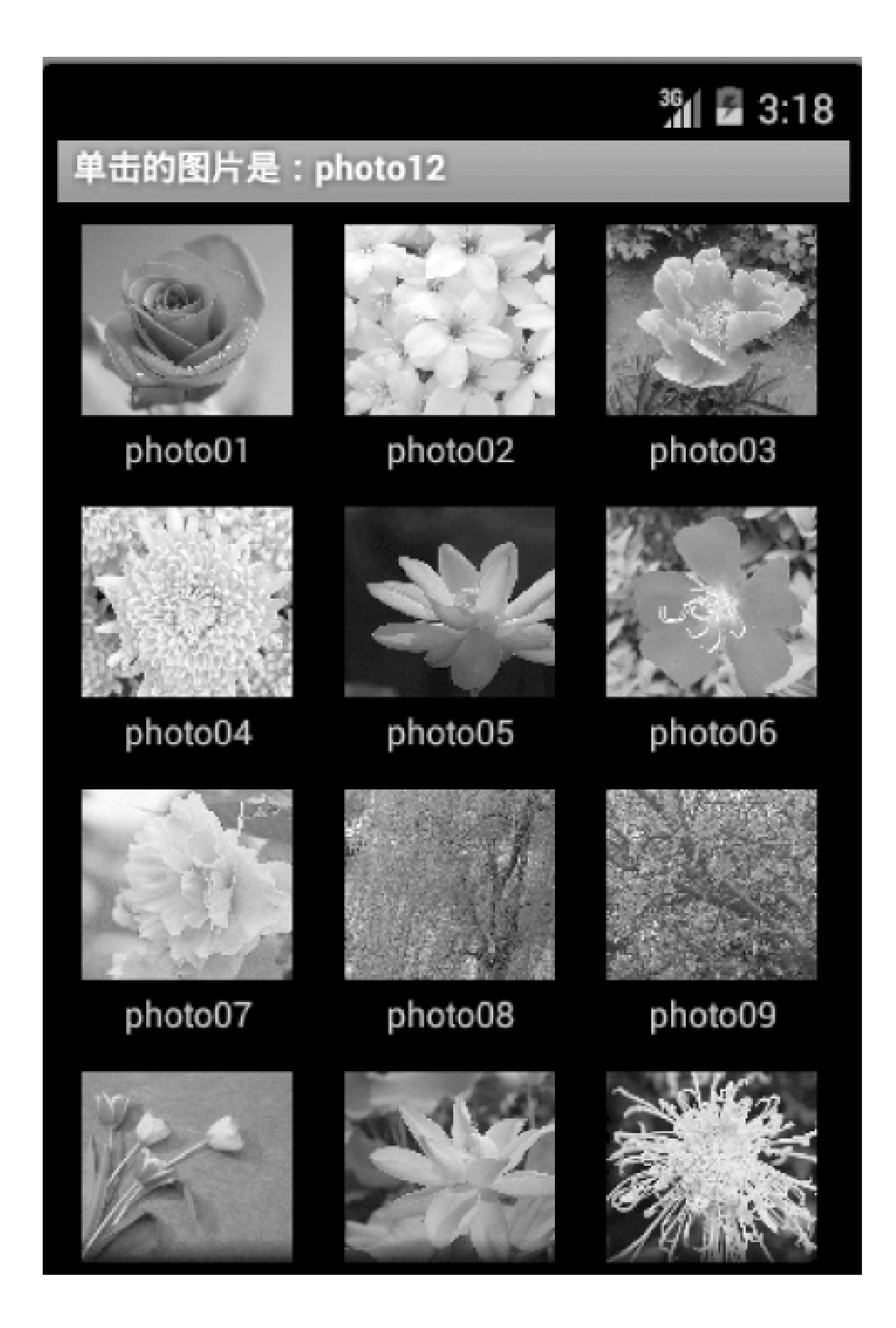


图 5.18 单击网格图片后显示的界面

5.2.6 ScrollView

在 ScrollView(滚动视图)中控件的内容在一屏显示不完时,便会自动产生滚动功能,通过纵向滚动的方式以显示被挡住的部分内容。

ScrollView 类位于 android. widget 包下,继承自 FrameLayout。ScrollView 只支持垂直滚动。ScrollView 中只能加一个控制,一般是嵌入一个线性布局。

5.2.7 TabHost

将 TabHost(选项卡)的 Tab 与 Host 拆开,易于理解 TabHost 指标签与宿主对应关系。由于标签内容对应于页面内容,通过一页中多个标签,可在一页中显示多个页面内容。所以,Tab 与 Host 对应关系为标签与页面对应关系。

TabHost 类位于 android. widget 包下,继承自 FrameLayout,是一种帧布局。如果它包含了多个布局,在同一时刻,只显示其中一个布局的内容。

【例 5.11】 创建选项卡,通过一页中不同的标签显示不同的页面内容。

【解题思路】

在一个帧布局 FrameLayout 中,定义多个线性布局 LinearLayout,在每个 LinearLayout 中添加页面图片和页面图片名称。

【开发步骤和程序分析】

(1) 在 Eclipse 中创建一个 TabHostExample 应用项目,包名为 com. application. tabhostexample。

(2) 图片资源,将图片资源复制到 res/drawable-mdpi 中,页面图片资源名为 photo01 . jpg～photo04. jpg,标签图片资源名为 photo01tab. jpg～photo04tab. jpg。

(3) 字符串资源,在 res/layout 目录下的 string. xml 文件中,编辑代码如下:

```
<?xml version = "1.0" encoding = "utf - 8"?>
<resources>
    <string name = "hello">Hello World, TabHostExample!</string>
    <string name = "app_name">TabHostExample</string>
    <string name = "photo01">梨花 \n
        </string>
    <string name = "photo02">柳枝 \n
         </string>
    <string name = "photo03">樱花 \n
         </string>
    <string name = "photo04">海棠花 \n
         </string>
</resources>
```

(4) 设计布局,在 res/layout 目录下的 main. xml 文件中,定义 FrameLayout 布局。在 FrameLayout 布局中,分别定义 4 个 LinearLayout 布局,其中的 ImageView 控件用于显示页面图片,TextView 控件用于显示页面图片标题。

在该文件中编辑代码如下：

```
<?xml version = "1.0" encoding = "utf - 8"?>
<! -- 定义 FrameLayout 布局 -->
< FrameLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent">
    <! -- 定义 LinearLayout 布局 -->
    < LinearLayout android:id = "@ + id/linearLayout01"
        android:layout_width = "fill_parent"  android:layout_height = "fill_parent"
        android:gravity = "center_horizontal"  android:orientation = "vertical">
       <! -- 设置 ImageView 控件 -->
       < ImageView
             android:id = "@ + id/tab_ImageView01"
             android:scaleType = "fitXY"
             android:layout_gravity = "center"
             android:layout_width = "wrap_content"
             android:layout_height = "wrap_content"
             android:src = "@drawable/photo01"/>
       <! -- 设置 TextView 控件 -->
       < TextView
             android:id = "@ + id/tab_TextView01"
             android:layout_width = "wrap_content"
             android:layout_height = "wrap_content"
             android:textSize = "24dip"
             android:textColor = "#f37301"
       android:text = "@string/photo01"/>
    </LinearLayout >
    < LinearLayout android:id = "@ + id/linearLayout02"
        android:layout_width = "fill_parent"  android:layout_height = "fill_parent"
        android:gravity = "center_horizontal"  android:orientation = "vertical" >
       < ImageView
             android:id = "@ + id/tab_ImageView02"
             android:scaleType = "fitXY"
             android:layout_gravity = "center"
             android:layout_width = "wrap_content"
             android:layout_height = "wrap_content"
             android:src = "@drawable/photo02"/>
         < TextView
             android:id = "@ + id/tab_TextView02"
             android:layout_width = "wrap_content"
             android:layout_height = "wrap_content"
             android:textSize = "24dip"
             android:textColor = "#f37301"
             android:text = "@string/photo02"/>
```

```
44   </LinearLayout>
45   <LinearLayout android:id="@+id/linearLayout03"
46         android:layout_width="fill_parent"  android:layout_height="fill_parent"
47         android:gravity="center_horizontal"  android:orientation="vertical">
48         <ImageView
49               android:id="@+id/tab_ImageView03"
50               android:scaleType="fitXY"
51               android:layout_gravity="center"
52               android:layout_width="wrap_content"
53               android:layout_height="wrap_content"
54               android:src="@drawable/photo03"/>
55           <TextView
56               android:id="@+id/tab_TextView03"
57               android:layout_width="wrap_content"
58               android:layout_height="wrap_content"
59               android:textSize="24dip"
60           android:textColor="#f37301"
61               android:text="@string/photo03"/>
62    </LinearLayout>
63    <LinearLayout android:id="@+id/linearLayout04"
64          android:layout_width="fill_parent"  android:layout_height="fill_parent"
65          android:gravity="center_horizontal"  android:orientation="vertical">
66         <ImageView
67               android:id="@+id/tab_ImageView04"
68               android:scaleType="fitXY"
69               android:layout_gravity="center"
70               android:layout_width="wrap_content"
71               android:layout_height="wrap_content"
72               android:src="@drawable/photo04"/>
73           <TextView
74               android:id="@+id/tab_TextView04"
75               android:layout_width="wrap_content"
76           android:layout_height="wrap_content"
77               android:textSize="24dip"
78               android:textColor="#f37301"
79               android:text="@string/photo04"/>
80    </LinearLayout>
81 </FrameLayout>
```

① 第 2 行至第 81 行定义 FrameLayout 布局。

② 第 7 行至第 26 行定义第一个 LinearLayout 布局，为垂直方向，ID 号为 linearLayout01。其中，ImageView 控件用于显示第一个页面图片，TextView 控件用于显示第一个页面图片标题。

③ 第 27 行至第 44 行、第 45 行至第 62 行、第 63 行至第 80 行，分别定义第 2 个、第 3 个、第 4 个 LinearLayout 布局，其中的控件分别用于显示相应的页面图片和页面图片标题。

(5) 在 src/com.application.tabhostexample 包下的 TabHostExampleActivity.java 文件中，定义 TabHostExampleActivity 类继承自 TabActivity 类，获取 TabHost 对象。在该对象中，分别设置第 1 个、第 2 个、第 3 个、第 4 个选项卡的标签图片名称和标签图片、页面图片名称和页面图片。

在该文件中编辑代码如下：

```
package com.application.tabhostexample;

import com.application.tabhostexample.R;
import android.app.TabActivity;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.widget.TabHost;

//定义 TabHostExampleActivity 类继承自 TabActivity 类
 public class TabHostExampleActivity extends TabActivity {
     private TabHost myTabhost;
     public void onCreate(Bundle savedInstanceState) {
         super.onCreate(savedInstanceState);
         //获取 TabHost 对象 myTabhost
         myTabhost = this.getTabHost();
         //从布局文件 main.xml 中获取显示模板
         LayoutInflater.from(this).inflate(R.layout.main, myTabhost.getTabContentView(), true);

         //设置第 1 个选项卡的内容
         myTabhost.addTab(
                     //在 myTabhost 上添加一个名为"选项卡 1"的选项卡
                     myTabhost.newTabSpec("选项卡 1")
                     //设置第 1 个标签图片名称和标签图片
                     .setIndicator("梨花",getResources().getDrawable(R.drawable.photo01tab))
                     //设置第 1 个页面图片名称和页面图片
                     .setContent(R.id.linearLayout01)
                     );

         myTabhost.addTab(
                     myTabhost.newTabSpec("选项卡 2")
                     .setIndicator("柳枝", getResources().getDrawable(R.drawable.photo02tab))
                     .setContent(R.id.linearLayout02)
                     );

         myTabhost.addTab(
                     myTabhost.newTabSpec("选项卡 3")
                     .setIndicator("樱花", getResources().getDrawable(R.drawable.photo03tab))
                     .setContent(R.id.linearLayout03)
                     );
```

```
40
41          myTabhost.addTab(
42                  myTabhost.newTabSpec("选项卡 4")
43                  .setIndicator("海棠花", getResources().getDrawable(R.drawable.photo04tab))
44                  .setContent(R.id.linearLayout04)
45                  );
46      }
47 }
```

① 第 10 行声明 TabHostExampleActivity 类继承自 TabActivity 类，为此，在第 4 行引入 import android.app.TabActivity 包。

② 第 15 行从 TabActivity 上面获取 TabHost 对象 myTabhost。

③ 第 17 行从布局文件 main.xml 中获取显示模板。

④ 第 20 行至第 27 行设置第 1 个选项卡的内容。其中，第 22 行在 myTabhost 上添加一个名为“选项卡 1”的选项卡；第 24 行设置标签(Tab)的内容，包括第 1 个标签图片名称“梨花”和第 1 个标签图片 photo01tab.jpg；第 26 行设置第 1 个页面(Host)显示的布局 ID 号为 linearLayout01，其中的控件显示第 1 个页面图片名称“梨花”和第 1 个页面图片 photo01.jpg。

⑤ 第 29 行至第 33 行、第 35 行至第 39 行、第 41 行至第 45 行，分别设置第 2 个选项卡、第 3 个选项卡、第 4 个选项卡的标签和页面的内容。

【运行结果】

在 Eclipse 中启动模拟器，然后运行项目 TabHostExample，选项卡选中“柳枝”标签后显示“柳枝”页面，如图 5.19 所示。

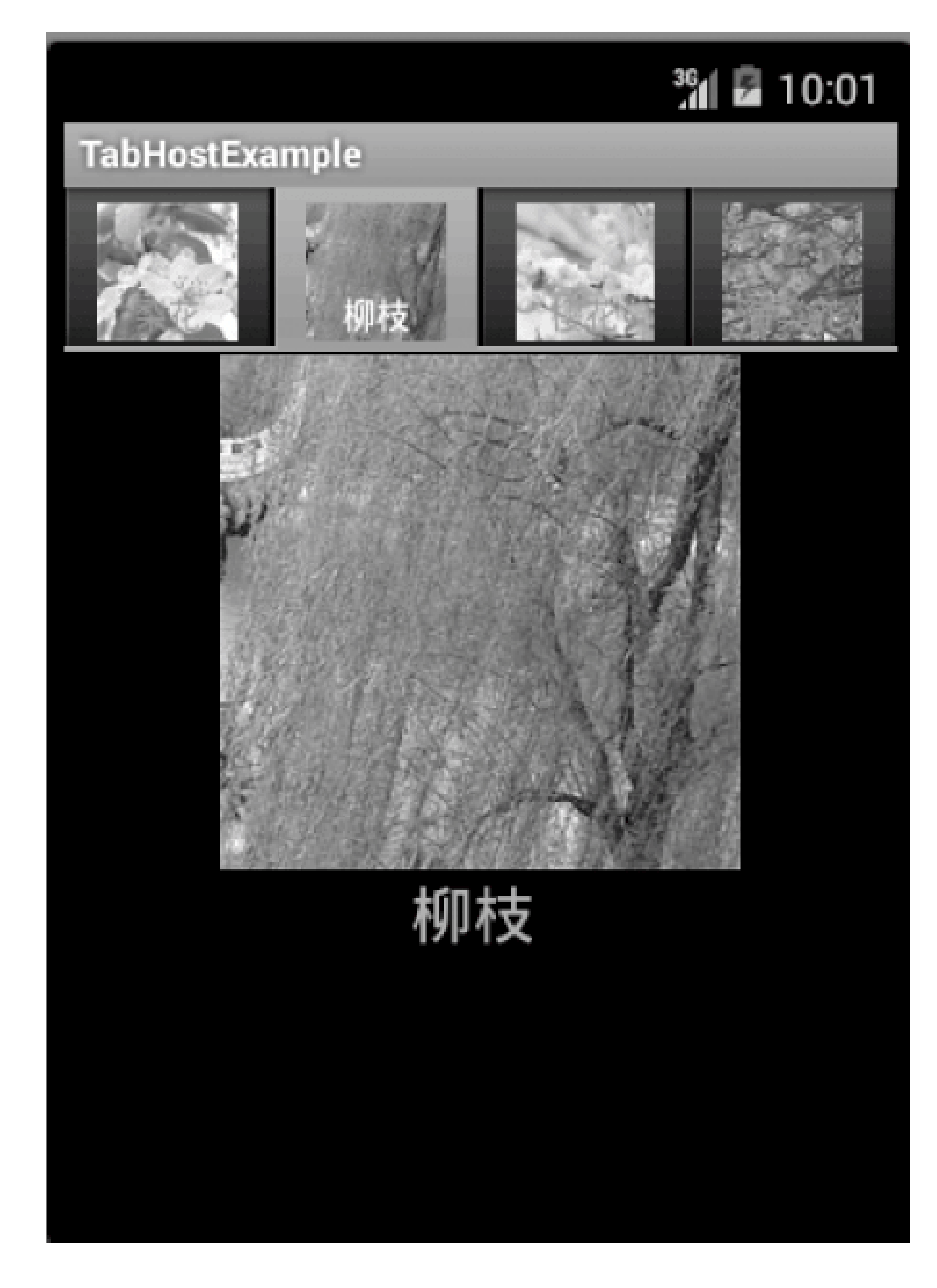


图 5.19　选项卡选中“柳枝”标签后显示”柳枝”页面

5.2.8　ImageSwitcher

ImageSwitcher(图像切换器)用于多张图片的切换，是 Android 中控制图片展示效果的一个控件。ImageSwitcher 控件常与 Gallery 控件一起使用。

常用方法有：

(1) setImageURI(Uri uri)，设置图片地址。

(2) setImageResource(int resid)，设置图片资源库。

(3) setImageDrawable(Drawable drawable)，绘制图片。

【例 5.12】 创建图像切换器，在屏幕下边是一组可以滚动的小图片，屏幕中部显示选

中的大图片。

【解题思路】

ImageSwitcher 作为大图片的展示控件，Gallery 作为小图片的滚动控件。

【开发步骤和程序分析】

(1) 在 Eclipse 中创建一个 ImageSwitcherExample 应用项目，包名为 com.application.imageswitcherexample。

(2) 准备资源，将图片资源复制到 res/drawable-mdpi 中，大图片资源名为 photo01.jpg～photo04.jpg，小图片资源名为 photo01sw.jpg～photo04sw.jpg。

(3) 设计布局，在 res/layout 目录下的 imain.xml 文件中，定义 RelativeLayout 布局。在该布局中，定义一个 ImageSwitcher 控件，用于显示选中的大图片；定义一个 Gallery 控件，用于显示横向滚动的小图片列表索引。

在该文件中编辑代码如下：

```
1  <?xml version = "1.0" encoding = "utf - 8"?>
2  <! -- 定义 RelativeLayout 布局 -->
3  < RelativeLayout
4          xmlns:android = "http://schemas.android.com/apk/res/android"
5          android:layout_width = "fill_parent"
6          android:layout_height = "fill_parent">
7          <! -- 定义一个 ImageSwitcher 控件,用于显示选中的大图片 -->
8          < ImageSwitcher
9                  android:id = "@ + id/switcher"
10                 android:layout_width = "fill_parent"
11                 android:layout_height = "fill_parent"
12                 android:layout_alignParentTop = "true"
13                 android:layout_alignParentLeft = "true" />
14         <! -- 定义一个 Gallery 控件,用于显示横向滚动的小图片列表索引 -->
15         < Gallery android:id = "@ + id/gallery"
16                 android:background = "#55000000"
17                 android:layout_width = "fill_parent"
18                 android:layout_height = "60dp"
19                 android:layout_alignParentBottom = "true"
20                 android:layout_alignParentLeft = "true"
21                 android:gravity = "center_vertical"
22                 android:spacing = "16dp"
23                 android:unselectedAlpha = "1"/>
24 </RelativeLayout >
```

① 第 3 行至第 24 行定义 RelativeLayout 布局。

② 第 8 行至第 13 行定义一个 ImageSwitcher 控件，用于显示选中的大图片。

③ 第 15 行至第 23 行定义一个 Gallery 控件，用于显示横向滚动的小图片列表索引。其中，第 16 行设置该 Gallery 背景颜色为半透明的黑色，当大图片尺寸较大与 Gallery 控件有重叠时，半透明背景色不会遮住大图片的显示。

(4) 在 src/com.application.imageswitcherexample 包下的 ImageSwitcherExampleActivity.java 文件中，定义 ImageSwitcherExampleActivity 类继承自 Activity 类，并实现接口 AdapterView.OnItemSelectedListener 和 ViewSwitcher.ViewFactory；设置 ImageSwitcher 对象及其切换效果，定义回调方法为 ImageSwitcher 对象获取选中的图片资源并显示，为 Gallery 对象绑定一个自定义的适配器和一个滚动监听。

在该文件中编辑代码如下：

```
package com.application.imageswitcherexample;

import com.application.imageswitcherexample.R;
import android.app.Activity;
import android.content.Context;
import android.os.Bundle;
import android.view.View;
import android.view.ViewGroup;
import android.view.Window;
import android.view.animation.AnimationUtils;
import android.widget.AdapterView;
import android.widget.BaseAdapter;
import android.widget.Gallery;
import android.widget.ImageSwitcher;
import android.widget.ImageView;
import android.widget.ViewSwitcher;
import android.widget.Gallery.LayoutParams;

//声明 ImageSwitcherExampleActivity 类继承自 Activity 类,
//并实现接口 AdapterView.OnItemSelectedListener 和 ViewSwitcher.ViewFactory
public class ImageSwitcherExampleActivity extends Activity implements
AdapterView.OnItemSelectedListener, ViewSwitcher.ViewFactory {

    private ImageSwitcher mySwitcher;
    //声明一个图片资源 ID 数组 galleryIds,为 Gallery 准备数据
    private Integer[] galleryIds = {
         R.drawable.photo01sw, R.drawable.photo02sw, R.drawable.photo03sw,
         R.drawable.photo04sw};
    //声明一个图片资源 ID 数组 myImgIds,为 ImageSwitcher 准备数据
    private Integer[] myImgIds = {
            R.drawable.photo01,R.drawable.photo02,R.drawable.photo03,
            R.drawable.photo04};

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        requestWindowFeature(Window.FEATURE_NO_TITLE);
```

```java
        setContentView(R.layout.imain);

        //设置 ImageSwitcher 对象 mySwitcher 及其切换效果
        mySwitcher = (ImageSwitcher) findViewById(R.id.switcher);
        mySwitcher.setFactory(this);
        mySwitcher.setInAnimation(AnimationUtils.loadAnimation(this,
                android.R.anim.fade_in));
        mySwitcher.setOutAnimation(AnimationUtils.loadAnimation(this,
                android.R.anim.fade_out));

        Gallery gallery = (Gallery) findViewById(R.id.gallery);
        //为 Gallery 对象 gallery 绑定一个自定义的 ImageAdapter 适配器
        gallery.setAdapter(new ImageAdapter(this));
        //为 Gallery 对象 gallery 绑定一个滚动监听 OnItemSelectedListener
        gallery.setOnItemSelectedListener(this);
    }

    //定义回调方法 onItemSelected(),为 ImageSwitcher 对象获取选中的图片资源并显示
    public void onItemSelected(AdapterView<?> parent, View v, int position, long id) {
        mySwitcher.setImageResource(myImgIds[position]);
    }

    public void onNothingSelected(AdapterView<?> parent) {
    }

    public View makeView() {
        ImageView iv = new ImageView(this);
        iv.setBackgroundColor(0xFF000000);
        iv.setScaleType(ImageView.ScaleType.FIT_CENTER);
        iv.setLayoutParams(new ImageSwitcher.LayoutParams(LayoutParams.FILL_PARENT,
                LayoutParams.FILL_PARENT));
        return iv;
    }

    //自定义 ImageAdapter 适配器
    public class ImageAdapter extends BaseAdapter {
        private Context myContext;
        public ImageAdapter(Context c) { myContext = c; }
        public int getCount() { return galleryIds.length; }
        public Object getItem(int position) { return position; }
        public long getItemId(int position) { return position; }
        //重写 getView()方法,为滚动的图片列表索引赋予资源数组 galleryIds 的值
        public View getView(int position, View convertView, ViewGroup parent) {
            ImageView i = new ImageView(myContext);
            i.setImageResource(galleryIds[position]);
```

```
83              i.setAdjustViewBounds(true);
84              i.setLayoutParams(new Gallery.LayoutParams(
85                      LayoutParams.WRAP_CONTENT, LayoutParams.WRAP_CONTENT));
86              return i;
87          }
88      }
89  }
```

① 第 21 行至第 89 行声明 ImageSwitcherExampleActivity 类继承自 Activity 类，并实现接口 AdapterView.OnItemSelectedListener 和接口 ViewSwitcher.ViewFactory。

② 第 26 行至第 28 行声明一个图片资源 ID 数组 galleryIds，为 Gallery 准备数据；第 30 行至第 32 行声明一个图片资源 ID 数组 myImgIds，为 ImageSwitcher 准备数据。

③ 第 37 行设置这个 Activity 标题栏不可见。

④ 第 41 行至第 46 行设置 ImageSwitcher 对象 mySwitcher。ImageSwitcher 的切换效果由第 43 行至第 46 行实现。

⑤ 第 50 行为 Gallery 对象 gallery 绑定一个自定义的 ImageAdapter 适配器。

⑥ 第 52 行为 Gallery 对象 gallery 绑定一个滚动监听 OnItemSelectedListener；第 56 行至第 58 行为该监听定义回调方法 onItemSelected()，为 ImageSwitcher 对象 mySwitcher 获取选中的图片资源并显示。

⑦ 第 73 行至第 88 行自定义 ImageAdapter 适配器。其中，第 80 行至第 87 行重写 getView() 方法，为滚动的图片列表索引赋予资源数组 galleryIds 的值。

【运行结果】

在 Eclipse 中启动模拟器，然后运行项目 ImageSwitcherExample，选中图像切换器小图片后显示对应的大图片，如图 5.20 所示。



图 5.20 选中图像切换器小图片后显示对应的大图片

5.2.9 进度条与拖动条

ProgressBar(进度条)是一种实用控件，通常用于加载资源或执行某些耗时操作。

SeekBar(拖动条)是一种滑块控件，继承自 ProgressBar，用来接收用户拖拉滑块操作，通常用于调节声音大小等操作。

RatingBar(星级评分条)是另一种滑块控件，外观是 5 个星星，通常用于星级评分等操作。

【例 5.13】 创建进度条、拖动条和星级评分条，当拖动条或星级评分条的滑块被拖动时，其他进度条也同步移动。

【解题思路】

为 ProgressBar 和 SeekBar 设置进度刻度，使用方法 setProgress(int)，获取其进度数据使用方法 getProgress()；为 RatingBar 设置星级，使用方法 setRating(float)，获取其星级使用方法 getRating()。

【开发步骤和程序分析】

(1) 在 Eclipse 中创建一个 ProgressBarsExample 应用项目，包名为 com. application. progressbarsexample。

(2) 设计布局，在 res/layout 目录下的 main. xml 文件中，定义垂直线性布局 LinearLayout。在该布局中，设置一个 ProgressBar 控件(进度条)、一个 SeekBar 控件(滑块进度条)、一个 RatingBar 控件(五星形滑块进度条)。

在该文件中编辑代码如下：

```
<?xml version = "1.0" encoding = "utf - 8"?>
<! -- 定义一个垂直线性布局 -->
<LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = "vertical"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent">
  <TextView
      android:id = "@ + id/Text01"
      android:layout_width = "fill_parent"
       android:layout_height = "wrap_content"
      android:text = "ProgressBar:"/>
    <! -- 设置一个 ProgressBar 控件 -->
    <ProgressBar
      android:id = "@ + id/ProgressBar01"
      android:layout_width = "fill_parent"
      android:layout_height = "wrap_content"
      android:max = "100"
      android:progress = "40"
      style = "@android:style/Widget.ProgressBar.Horizontal"/>
    <TextView
      android:id = "@ + id/Text02"
      android:layout_width = "fill_parent"
      android:layout_height = "wrap_content"
      android:text = "SeekBar:"/>
  <! -- 设置一个 SeekBar 控件 -->
  <SeekBar
      android:id = "@ + id/SeekBar01"
      android:layout_width = "fill_parent"
      android:layout_height = "wrap_content"
      android:max = "100"
      android:progress = "40"/>
```

```
32      <TextView
33        android:id="@+id/Text03"
34        android:layout_width="fill_parent"
35        android:layout_height="wrap_content"
36        android:text="RatingBar:"/>
37     <!-- 设置一个 RatingBar 控件 -->
38     <RatingBar
39        android:id="@+id/RatingBar01"
40        android:layout_width="wrap_content"
41        android:layout_height="wrap_content"
42        android:max="5"
43        android:rating="2"
44     />
45 </LinearLayout>
```

① 第 3 行至第 45 行定义一个垂直线性布局 LinearLayout。

② 第 13 行至第 19 行设置一个 ProgressBar 控件，为进度条。

③ 第 26 行至第 31 行设置一个 SeekBar 控件，为滑块进度条。

④ 第 38 行至第 44 行设置一个 RatingBar 控件，为五星形滑块进度条。

(3) 在 src/com.application.progressbarsexample 包下的 ProgressBarsExampleActivity.java 文件中，定义 ProgressBarsExampleActivity 类继承自 Activity 类，设置 SeekBar 的拖拉监听(当 SeekBar 被拖拉监听时，同步设置 ProgressBar、SeekBar 的值)，设置 RatingBar 的拖拉监听(当 RatingBar 被拖拉监听时，同步设置 ProgressBar、SeekBar 的值)。

在该文件中编辑代码如下：

```
package com.application.progressbarsexample;

import com.application.progressbarsexample.R;
import android.app.Activity;
import android.os.Bundle;
import android.widget.ProgressBar;
import android.widget.RatingBar;
import android.widget.SeekBar;

public class ProgressBarsExampleActivity extends Activity {
    //定义常量 MAX 值为 100,作为 SeekBar、ProgressBar 的最大值
    final static double MAX = 100;
    //定义常量 MAX_STAR 值为 5,作为 RatingBar 的最大星星数
    final static double MAX_STAR = 5;
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
       //设置 SeekBar 的拖拉监听,当 SeekBar 被拖拉监听时,
       //同步设置 ProgressBar、SeekBar 的值
```

```
20          SeekBar sb = (SeekBar)this.findViewById(R.id.SeekBar01);
21          sb.setOnSeekBarChangeListener(
22              new SeekBar.OnSeekBarChangeListener(){
23                  public void onProgressChanged(SeekBar seekBar, int progress,
24                          boolean fromUser) {
25                      ProgressBar pb = (ProgressBar)findViewById(R.id.ProgressBar01);
26                      RatingBar rb = (RatingBar)findViewById(R.id.RatingBar01);
27                      SeekBar sb = (SeekBar)findViewById(R.id.SeekBar01);
28                      pb.setProgress(sb.getProgress());
29                      rb.setRating((float)(sb.getProgress()/MAX * MAX_STAR));
30                  }
31                  public void onStartTrackingTouch(SeekBar seekBar) {     }
32                  public void onStopTrackingTouch(SeekBar seekBar) { }
33              }
34          );
35
36          RatingBar rb = (RatingBar)findViewById(R.id.RatingBar01);
37          //设置 RatingBar 的拖拉监听,当 RatingBar 被拖拉监听时,
38          //同步设置 ProgressBar、SeekBar 的值
39          rb.setOnRatingBarChangeListener(
40                  new RatingBar.OnRatingBarChangeListener(){
41                      @Override
42                      public void onRatingChanged(RatingBar ratingBar, float rating,
43                              boolean fromUser) {
44                          ProgressBar pb = (ProgressBar)findViewById(R.id.ProgressBar01);
45                          SeekBar sb = (SeekBar)findViewById(R.id.SeekBar01);
46                          RatingBar rb = (RatingBar)findViewById(R.id.RatingBar01);
47                          float rate = rb.getRating();
48                          pb.setProgress((int) (rate/MAX_STAR * MAX));
49                          sb.setProgress((int) (rate/MAX_STAR * MAX));
50                      }
51                  }
52          );
53      }
54 }
```

① 第 12 行定义常量 MAX 值为 100,作为 SeekBar、ProgressBar 的最大值;第 14 行定义常量 MAX_STAR 值为 5,作为 RatingBar 的最大星星数。

② 第 20 行至第 34 行设置 SeekBar 的拖拉监听。当 SeekBar 被拖拉监听时,同步设置 ProgressBar、SeekBar 的值。

③ 第 39 行至第 52 行设置 RatingBar 的拖拉监听。当 RatingBar 被拖拉监听时,同步设置 ProgressBar、SeekBar 的值。

【运行结果】

在 Eclipse 中启动模拟器,然后运行项目 ProgressBarsExample,运行结果如图 5.21 所示。

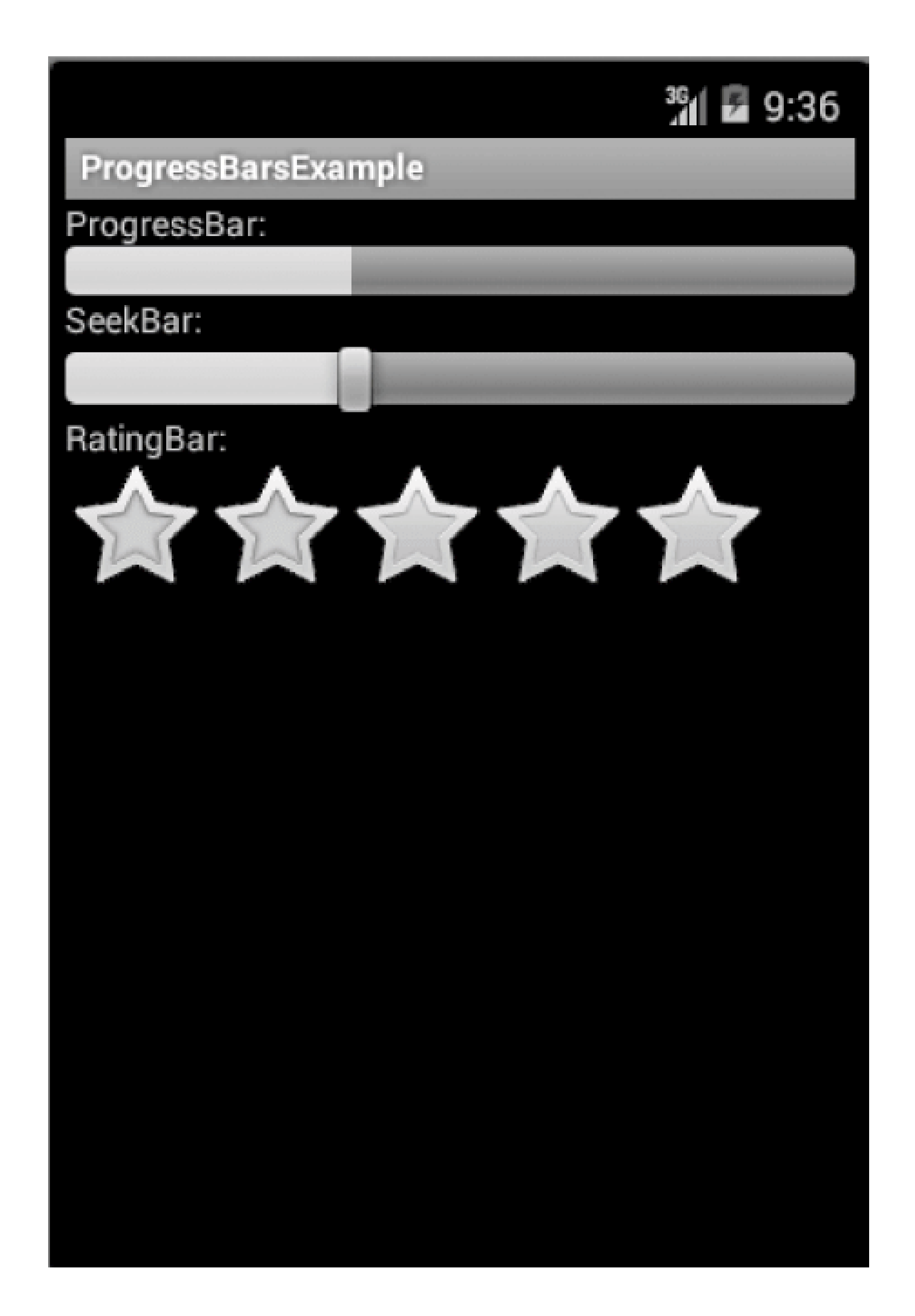


图 5.21　进度条、拖动条和星级评分条

5.2.10　应用项目的界面设计

下面通过综合实例——网上购计算机应用项目的界面设计，进一步理解和掌握本章介绍的高级控件内容。

【例 5.14】 网上购计算机应用项目的界面设计。

【解题思路】

使用 TabHost 控件来设计网上购计算机中心框架界面，根据网上购计算机功能模块分类来确定选项卡标签，然后在相应的页面中显示相应的 Activity。

【开发步骤和程序分析】

(1) 功能设计。网上购计算机的功能模块有"新品上市""笔记本计算机""台式计算机""服务器"等 4 个模块，在"新品上市"模块界面中，用列表形式显示笔记本计算机新品。

(2) 界面设计(UI 设计)。根据网上购计算机功能可使用 TabHost 控件进行界面设计，按"新品""笔记本""台式机""服务器"的顺序添加 Tab。其中，第一个 Tab 为默认显示的界面，即"新品"界面，其页面可使用 ListView 控件将笔记本计算机新品一一列出。本例只详细设计这个页面，其余页面以后补充。

(3) 创建项目。在 Eclipse 中创建一个 OnlineShoppingComputer 应用项目，包名为 com. application. onlineshoppingcomputer。

(4) 准备图片资源，将图片资源复制到 res/drawable-mdpi 中。

(5) 准备字符串资源，在 res/values 目录下的 string. xml 文件中，编辑代码如下：

```
<?xml version = "1.0" encoding = "utf - 8"?>
< resources >
    < string name = "hello"> Hello World, OnlineShopping!</string >
    < string name = "app_name"> OnlineShopping </string >
    < string name = "OnlineShopping">网上购计算机</string >
    < string name = "PublishActivity">网上购计算机 -- 新品</string >
    < string name = "ContactsActivity">网上购计算机 -- 笔记本</string >
    < string name = "MyDiaryActivity">网上购计算机 -- 台式机</string >
    < string name = "AlbumListActivity">网上购计算机 -- 服务器</string >
    < string name = "btnBack">返回</string >
    < string name = "tabtitle1">新品</string >
    < string name = "tabtitle2">笔记本</string >
    < string name = "tabtitle3">台式机</string >
    < string name = "tabtitle4">服务器</string >
</resources >
```

(6) 创建数组资源,在 res/values 目录下的 arrays. xml 文件中,编辑代码如下:

```
<?xml version = "1.0" encoding = "utf - 8"?>
< resources >
    < string - array name = "lvtexts">
        < item >戴尔</item >
        < item >联想</item >
    </string - array >
    < string - array name = "lvicons">
        < item > dell </item >
        < item > lenovo </item >
     </string - array >
</resources >
```

(7) 创建颜色资源,在 res/values 目录下的 colors. xml 文件中,编辑代码如下:

```
<?xml version = "1.0" encoding = "utf - 8"?>
< resources >
    < color name = "listDivider"> # ffcc99 </color >
    < color name = "character"> # f37301 </color >
</resources >
```

(8) 创建样式资源,在 res/values 目录下的 style. xml 文件中,编辑代码如下:

```
<?xml version = "1.0" encoding = "utf - 8"?>
< resources >
    < style name = "button">
        < item name = "android:textSize"> 20sp </item >
        < item name = "android:textColor"> # f37301 </item >
        < item name = "android:textStyle"> bold </item >
    </style >
```

```
    <style name = "title">
        <item name = "android:textSize">22sp</item>
        <item name = "android:textColor">#f37301</item>
        <item name = "android:textStyle">bold</item>
    </style>
    <style name = "text">
        <item name = "android:textSize">18sp</item>
        <item name = "android:textColor">#f37301</item>
        <item name = "android:textStyle">bold</item>
    </style>
</resources>
```

(9) 设计布局，在 res/layout 目录下的布局文件有 newarrival. xml、notebook. xml、desktop. xml、server. xml，这里仅介绍其中 newarrival. xml 和 notebook. xml 的代码。

在 newarrival. xml 文件中，定义垂直线性布局。在该布局中，设置一个 ListView 控件。在该文件中编辑代码如下：

```
1  <?xml version = "1.0" encoding = "utf - 8"?>
2  <! -- 定义一个垂直线性布局 LinearLayout -->
3  <LinearLayout
4      xmlns:android = "http://schemas.android.com/apk/res/android"
5      android:orientation = "vertical"
6      android:background = "@drawable/flower"
7      android:layout_width = "fill_parent"
8      android:layout_height = "fill_parent"
9      >
10     <! -- 设置一个 ListView 控件 -->
11     <ListView
12         android:id = "@ + id/lvPublish"
13         android:divider = "@color/listDivider"
14         android:dividerHeight = "2px"
15         android:layout_width = "fill_parent"
16         android:layout_height = "fill_parent"
17         />
18 </LinearLayout>
```

① 第 3 行至第 18 行定义一个垂直线性布局 LinearLayout。

② 第 11 行至第 17 行设置一个 ListView 控件。其中，第 13 行至第 14 行为列表视图控件的分隔线设置颜色和线的高度。

在 notebook. xml 文件中，定义相对布局 RelativeLayout。在该布局中，设置一个 ScrollView 控件(其中设置一个 TextView 控件)，设置一个 Button 控件。在该文件中编辑代码如下。

```
<?xml version = "1.0" encoding = "utf - 8"?>
<! -- 定义一个相对布局 RelativeLayout -->
```

```
<RelativeLayout
    xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = "vertical"
    android:paddingLeft = "8px"
    android:background = "@drawable/flower"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    >
    <!-- 设置一个 ScrollView 控件 -->
    <ScrollView
        android:fillViewport = "true"
        android:layout_width = "fill_parent"
        android:layout_height = "fill_parent"
        android:layout_alignParentTop = "true"
        android:layout_alignParentLeft = "true"
        >
        <!-- 设置一个 TextView 控件 -->
        <TextView
            android:id = "@ + id/txtv"
            android:layout_width = "fill_parent"
            android:layout_height = "fill_parent"
            android:textColor = "@color/character"
            android:text = " 笔记本页面建设中..."
            android:textSize = "20dip"
           />
       </ScrollView>
       <!-- 设置一个 Button 控件 -->
       <Button
       android:id = "@ + id/Dia_btnBack"
       style = "@style/button"
       android:layout_width = "120px"
       android:layout_height = "wrap_content"
       android:layout_alignParentBottom = "true"
       android:layout_alignParentRight = "true"
       android:layout_marginRight = "5px"
       android:text = "@string/btnBack"
       />
</RelativeLayout>
```

(10) 在 src/com. application. onlineshoppingcomputer 包下有 Tab. java 文件、NewArrivalActivity. java 文件、NotebookActivity. java 文件、DesktopActivity. java 文件、ServerActivity. java 文件。其中，Tab. java 文件是网上购电脑应用项目的中心框架部分，使用 TabHost 控件进行界面设计，NewArrivalActivity. java 文件是 TabHost 中第一个 Tab 对应的 Activity。下面仅介绍 Tab. java 文件和 NewArrivalActivity. java 文件的代码。

在 Tab. java 文件中，定义 Tab 类继承自 TabActivity 类，声明 4 个 Intent 对象，分别绑

定 4 个 Activity(NewArrivalActivity、NotebookActivity、DesktopActivity、ServerActivity),逐个向 TabHost 对象添加选项卡,从第 1 个到第 4 个,分别设置标签文本、标签图片和页面内容。

在该文件中编辑代码如下:

```
package com.application.onlineshoppingcomputer;

importcom.application.onlineshoppingcomputer.R;
import android.app.TabActivity;
import android.content.Intent;
import android.os.Bundle;
import android.view.Window;
import android.widget.TabHost;

public class Tab extends TabActivity{

   //重写 onCreate()方法
   @Override
   protected void onCreate(Bundle savedInstanceState) {
       super.onCreate(savedInstanceState);
       requestWindowFeature(Window.FEATURE_NO_TITLE);
       final TabHost tabHost = getTabHost();
       //以下 4 行,声明 4 个 Intent 对象,分别绑定 4 个 Activity
       Intent itNormal = new Intent(this,NewArrivalActivity.class);
       Intent itContact = new Intent(this,NotebookActivity.class);
       Intent itDiary = new Intent(this,DesktopActivity.class);
       Intent itAlbum = new Intent(this,ServerActivity.class);

       //逐个向 TabHost 对象 tabHost 添加选项卡,从第 1 个到第 4 个
       //添加名为 tab1 的选项卡,设置标签文本、标签图片和页面内容
       tabHost.addTab(tabHost.newTabSpec("tab1")
               .setIndicator(getResources().getText(R.string.tabtitle1),
                       getResources().getDrawable(R.drawable.allinone))
               .setContent(itNormal)
               );

       //添加名为 tab2 的选项卡,设置标签文本、标签图片和页面内容
       tabHost.addTab(tabHost.newTabSpec("tab2")
               .setIndicator(getResources().getText(R.string.tabtitle2),
                       getResources().getDrawable(R.drawable.notebook))
               .setContent(itContact)
               );

       //添加名为 tab3 的选项卡,设置标签文本、标签图片和页面内容
       tabHost.addTab(tabHost.newTabSpec("tab3")
```

```
41                  .setIndicator(getResources().getText(R.string.tabtitle3),
42                          getResources().getDrawable(R.drawable.desktop))
43                  .setContent(itDiary)
44                  );
45
46          //添加名为 tab4 的选项卡,设置标签文本、标签图片和页面内容
47          tabHost.addTab(tabHost.newTabSpec("tab4")
48                  .setIndicator(getResources().getText(R.string.tabtitle4),
49                          getResources().getDrawable(R.drawable.server))
50                  .setContent(itAlbum)
51                  );
52      }
53 }
```

① 第 16 行设置窗口标题栏不显示。

② 第 19 行至第 22 行,声明 4 个 Intent 对象,分别绑定 4 个 Activity。

③ 第 26 行至第 51 行,逐个向 TabHost 对象 tabHost 添加选项卡。其中,第 26 行至第 30 行添加名为 tab1 的选项卡,在 Indicator 中使用 getResources().getText(R.string.tabtitle1)设置标签文本,使用 getResources().getDrawable(R.drawable.allinone)设置标签图片,使用 setContent(itNormal)设置页面内容,这里是 itNormal,即显示 NewArrivalActivity 界面。

在 NewArrivalActivity.java 文件中,定义 NewArrivalActivity 类继承自 Activity 类,获取 ListView 的 item 项文本内容数组和图标资源数组,重写 BaseAdapter 的 getView()方法,定义 ListView 对象 lvPublish 每一个 item 的显示及内容,将适配器添加到 ListView 对象 lvPublish,为 lvPublish 对象添加 OnItemClickListener()监听,当单击 lvPublish 中的条目时,会跳转到相应的 Activity 中。在该文件中编辑代码如下:

```
package com.application.onlineshoppingcomputerg;

import com.application.onlineshoppingcomputer.R;
import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.Gravity;
import android.view.View;
import android.view.ViewGroup;
import android.view.ViewGroup.LayoutParams;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.BaseAdapter;
import android.widget.ImageView;
import android.widget.LinearLayout;
import android.widget.ListView;
import android.widget.TextView;

```

```
//定义 NewArrivalActivity 类继承自 Activity 类
public class NewArrivalActivity extends Activity{

    //重写 onCreate()方法
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.newarrival);
        ListView lvPublish = (ListView)findViewById(R.id.lvPublish);

        BaseAdapter baseadapter = new BaseAdapter() {
            //获取 ListView 的 item 项文本内容数组
            String[] items = getResources().getStringArray(R.array.lvtexts);
            //获取 ListView 的 item 项图标资源数组
            String[] imgIds = getResources().getStringArray(R.array.lvicons);

            //重写 BaseAdapter 的 getView()方法,
            //定义 ListView 对象 lvPublish 每一个 item 的显示及内容
            @Override
            public View getView(int position, View convertView, ViewGroup parent) {
                LinearLayout ll = new LinearLayout(NewArrivalActivity.this);
                ll.setOrientation(LinearLayout.HORIZONTAL);
                ll.setGravity(Gravity.CENTER);
                ImageView iv = new ImageView(NewArrivalActivity.this);
                iv.setAdjustViewBounds(true);
                iv.setImageDrawable(getResources().getDrawable(
                    getResources().getIdentifier(imgIds[position],"drawable",getPackageName())));
                ll.addView(iv);
                TextView tv = new TextView(NewArrivalActivity.this);
                tv.setPadding(10, 0, 0, 0);
                tv.setLayoutParams(new LinearLayout
                LayoutParams(LayoutParams.WRAP_CONTENT,
                    LayoutParams.WRAP_CONTENT));
                tv.setTextAppearance(NewArrivalActivity.this, R.style.title);
                tv.setText(items[position]);
                ll.addView(tv);
                return ll;
            }

            @Override
            public long getItemId(int position) {
                return 0;
            }

            @Override
```

```
64              public Object getItem(int position) {
65                  return null;
66              }
67
68              @Override
69              public int getCount() {
70                  return items.length;
71              }
72          };
73
74          //将适配器添加到 ListView 对象 lvPublish
75          lvPublish.setAdapter(baseadapter);
76          //为 lvPublish 对象添加 OnItemClickListener()监听,
77          //当单击了 lvPublish 中的条目时,会跳转到相应的 Activity 中
78          lvPublish.setOnItemClickListener(new OnItemClickListener() {
79
80              @Override
81              public void onItemClick(AdapterView<?> parent, View v, int position,
82                      long id) {
83                  switch(position){
84                  case 0:
85                      Intent intentDia = new Intent();
86                      intentDia.setClass(NewArrivalActivity.this, NotebookActivity.class);
87                      startActivity(intentDia);
88                      break;
89                  case 1:
90                      Intent intentAlb = new Intent();
91                      intentAlb.setClass(NewArrivalActivity.this, NotebookActivity.class);
92                      startActivity(intentAlb);
93                      break;
94                  }
95              }
96          });
97      }
98  }
```

① 第 31 行取得 ListView 的 item 项文本内容数组,第 33 行取得 ListView 的 item 项图标资源数组。

② 第 37 行至第 56 行重写 BaseAdapter 的 getView()方法,定义 ListView 对象 lvPublish 每一个 item 的显示及内容。

③ 第 78 行至第 96 行,为 lvPublish 对象添加 OnItemClickListener()监听,当单击 lvPublish 中的条目时,会跳转到相应的 Activity 中。

【运行结果】

在 Eclipse 中启动模拟器,然后运行项目 OnlineShoppingComputer,网上购计算机的初始界面如图 5.22 所示。



图 5.22　网上购计算机的初始界面

在初始界面单击“戴尔”后的界面如图 5.23 所示；在初始界面选择“服务器”选项卡后的界面如图 5.24 所示。



图 5.23　在初始界面单击“戴尔”后的界面



图 5.24　在初始界面选择“服务器”选项卡后的界面

5.3 菜　　单

菜单(Menu)设计在人机交互中是十分重要的。它提供了不同功能分组展示能力,应用十分广泛。

Android 平台下有三类菜单:选项菜单(OptionMenu)、子菜单(Submenu)、上下文菜单(ContentMenu)。

在 Android 中通过回调方法创建菜单并处理菜单按下的事件。

Android 系统的菜单支持如图 5.25 所示。

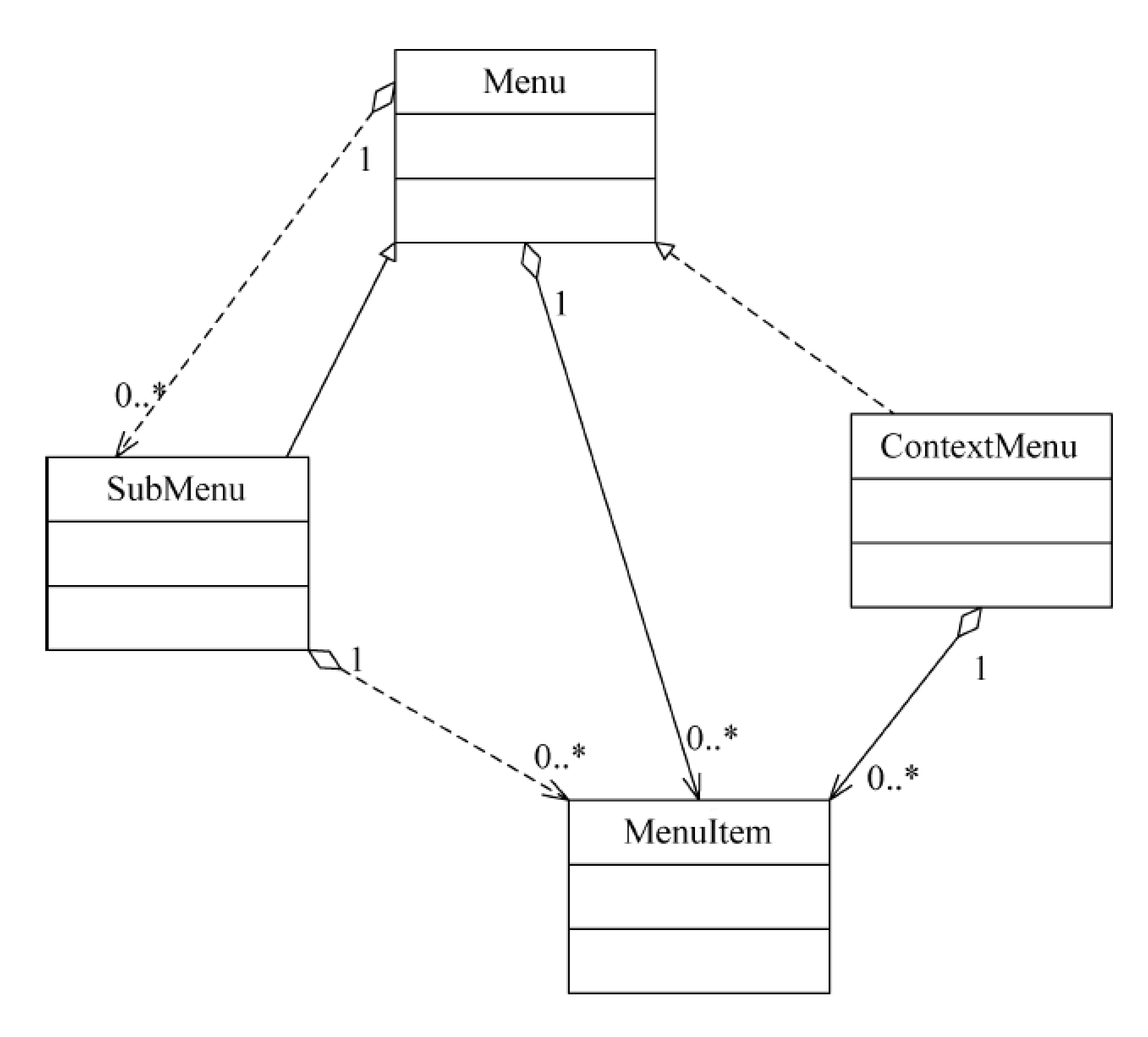


图 5.25　Android 系统的菜单支持

由图 5.25 可以看出,Android 系统的菜单支持主要通过 4 个接口来体现。Menu 接口是一个父接口,它有两个子接口 Submenu 和 ContentMenu。Submenu 代表一个子菜单,可以包含 1～N 个 MenuItem(形成菜单项)。ContentMenu 代表一个上下文菜单,可以包含 1～N 个 MenuItem(形成菜单项)。

5.3.1 选项菜单

Android 的选项菜单(OptionMenu)默认是看不见的,当用户按下 Menu 键时,选项菜单出现在屏幕下方。单击模拟器右边的 Menu 键,在屏幕底端弹出选项菜单,如图 5.26 所示。

选项菜单最多显示 6 个。当菜单选项多于 6 个时,只显示前 5 个菜单,最后的菜单项为一个扩展菜单选项。

选项菜单 OptionMenu 定义在 android.view.Menu 包中。一个选项菜单是一个 Menu 对象,在 Menu 对象中可以添加菜单项 MenuItem。

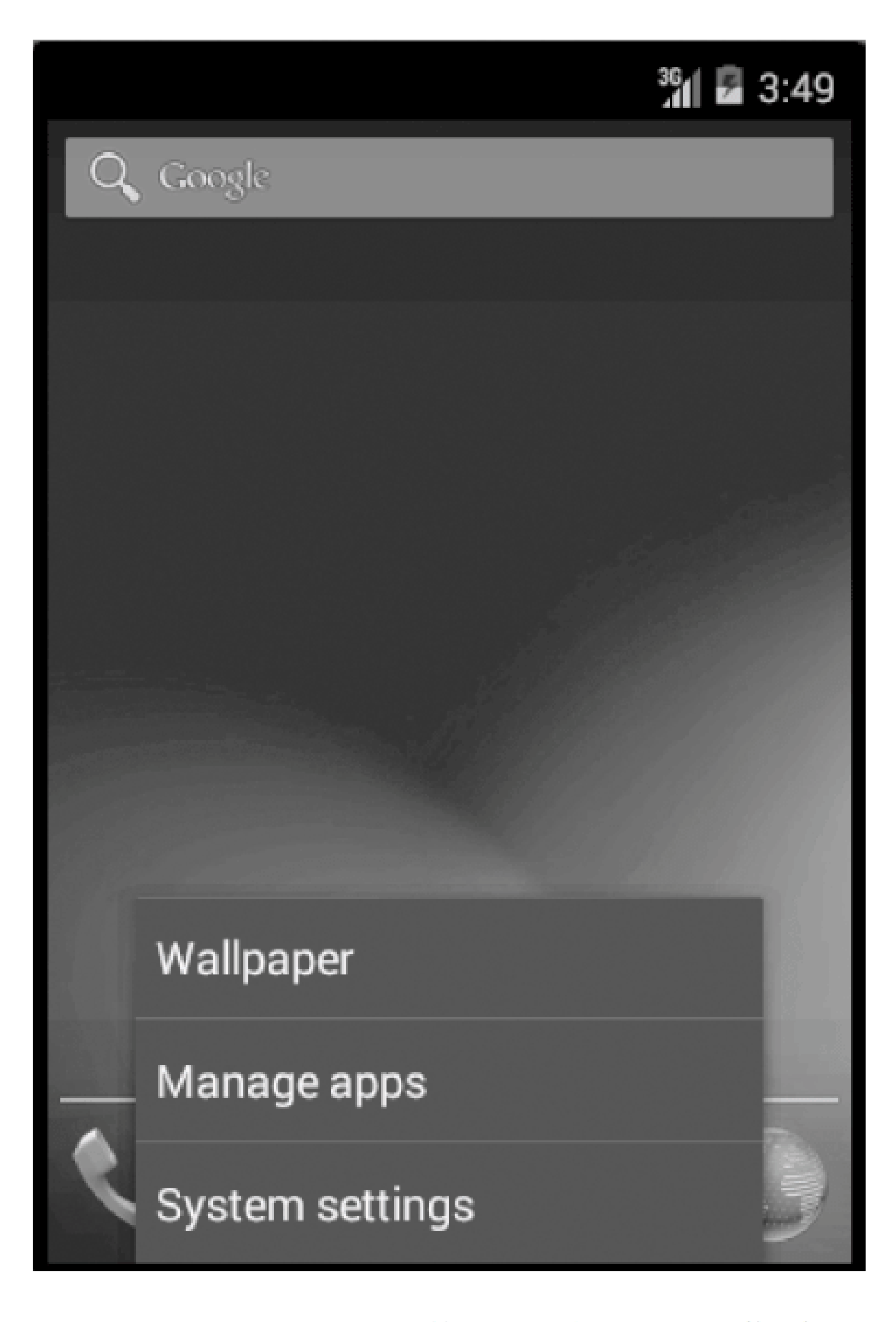


图 5.26　Android 模拟器自带选项菜单

选项菜单的功能需要开发人员编程来实现。开发选项菜单主要用到的类有 Menu、SubMenu、MenuItem。

选项菜单常用的回调方法如表 5.4 所示。

表 5.4　选项菜单常用的回调方法

方　　法	说　　明
onCreateOptionsMenu()	初始化选项菜单,只在首次显示菜单时调用
onOptionsItemSelected()	当某菜单项被选中时调用,默认返回 false
onOptionsMenuClosed()	当选项菜单关闭,或按下返回键,或选择了某菜单项时调用
onPrepareOptionsMenu()	为程序准备选项菜单,每次选项菜单显示前调用

Menu 类的对象是一个菜单,包含一个或多个菜单项 MenuItem,也可以包含子菜单 SubMenu。Menu 常用方法如表 5.5 所示。

表 5.5　Menu 常用方法

方　　法	说　　明
add()	向 Menu 添加一个菜单项,返回 MenuItem 对象
addSubMenu()	向 Menu 添加一个子菜单,返回 SubMenu 对象
finditem()	返回指定 id 的 MenuItem 对象
size()	返回 Menu 中菜单项的个数

MenuItem 常用方法如表 5.6 所示。

设计一个选项菜单时要为用户提供交互接口,以响应菜单项被单击的事件。

表 5.6　MenuItem 常用方法

方　　法	说　　明
setAlphabeticShortcut()	设置 MenuItem 的字母快捷键
setNumericShortcut()	设置 MenuItem 的数字快捷键
setIcon()	设置 MenuItem 的图标
setIntent()	为 MenuItem 绑定 Intent 对象,当被选中时调用 startActivity 方法处理动作相应的 Intent
setOnMenuItemClickListener()	为 MenuItem 设置自定义监听器。一般情况下,使用回调方法 onOptionsItemSelected 效率更高

创建选项菜单步骤如下:

(1) 重写 Activity 的 onCreateOptionsMenu(Menu menu)方法,第一次打开菜单时该方法被自动调用。

(2) 调用 Menu 的 add()方法添加菜单项 MenuItem。此时,可以调用 MenuItem 的 setIcon()方法来为菜单项设置图标。

(3) 定义菜单项被选择之后的回调事件。有两种方法:其一,重写 Activity 的 onOptionsItemSelected()方法,当菜单项 MenuItem 被选择时,该方法用于响应事件;其二,为每个菜单项 MenuItem 对象添加 OnMenuItemClickListener 监听器,在其中定义处理菜单选项中的事件。

【例 5.15】 创建选项菜单,当选择菜单项时,在文本框中显示选择的内容。

【解题思路】

使用为 MenuItem 对象添加 OnMenuItemClickListener 监听器的方式处理选择事件。

【开发步骤和程序分析】

(1) 在 Eclipse 中创建一个 OptionMenuExample 应用项目,包名为 com.application.optionmenuexample。

(2) 设计布局,在 res/layout 目录下的 main.xml 文件中,定义垂直线性布局。在该布局中设置一个 TextView 控件。在该文件中编辑代码如下:

```
<?xml version = "1.0" encoding = "utf - 8"?>
<! -- 定义一个垂直线性布局 LinearLayout -->
<LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    android:orientation = "vertical" >
    <! -- 设置一个 TextView 控件 -->
    <TextView
        android:id = "@ + id/tv01"
```

```
10          android:layout_width = "fill_parent"
11          android:layout_height = "wrap_content"
12          android:textSize = "20px"
13          android:text = "请选择..." />
14 </LinearLayout>
```

① 第 2 行至第 14 行定义一个垂直线性布局 LinearLayout。

② 第 9 行至第 13 行设置一个 TextView 控件。

(3) 在 com.application.optionmenuexample 包下的 OptionMenuExampleActivity.java 文件中,定义 OptionMenuExampleActivity 类继承自 Activity 类,定义 onCreateOptionsMenu()方法,在其中为选项菜单添加两个菜单项,创建 OnMenuItemClickListener 监听对象,定义回调方法 OnMenuItemClick(),为两个菜单项添加 OnMenuItemClickListener。在该文件中编辑代码如下:

```
package com.application.optionmenuexample;

import com.application.optionmenuexample.R;
import android.app.Activity;
import android.os.Bundle;
import android.widget.TextView;
import android.view.Menu;
import android.view.MenuItem;
import android.view.MenuItem.OnMenuItemClickListener;

//定义 OptionMenuExampleActivity 类继承自 Activity 类
public class OptionMenuExampleActivity extends Activity {
    private TextView text;
    //定义菜单项 ID 常量
    private static final int ITEM1 = Menu.FIRST;
    private static final int ITEM2 = Menu.FIRST + 1;

     @Override
     public void onCreate(Bundle savedInstanceState) {
         super.onCreate(savedInstanceState);
         setContentView(R.layout.main);
         text = (TextView) findViewById(R.id.tv01);
     }

    //定义 onCreateOptionsMenu()方法
    public boolean onCreateOptionsMenu(Menu menu) {
         //添加菜单项 begmenuitem,其 ID 号为 ITEM1,显示文本为"运动",并设置了图标
         MenuItem begmenuitem  =  menu.add(0, ITEM1, 0, "运动");
         begmenuitem.setIcon(R.drawable.sport);
         //添加菜单项 retmenuitem,其 ID 号为 ITEM1,显示文本为"音乐",并设置了图标
         MenuItem retmenuitem  =  menu.add(0, ITEM2, 0, "音乐");
```

```
32          retmenuitem.setIcon(R.drawable.music);
33
34          //创建 OnMenuItemClickListener 监听对象 lsn
35          OnMenuItemClickListener lsn = new OnMenuItemClickListener(){
36              @Override
37              public boolean onMenuItemClick(MenuItem item){
38                  switch (item.getItemId()) {
39                  case ITEM1:
40                      text.setText("你选择了: 运动!");
41                      break;
42                  case ITEM2:
43                      text.setText("你选择了: 音乐!");
44                      break;
45                  }
46                  return true;
47              }
48          };
49          //分别为两个菜单项添加 OnMenuItemClickListener
50          begmenuitem.setOnMenuItemClickListener(lsn);
51          retmenuitem.setOnMenuItemClickListener(lsn);
52          return true;
53      }
54 }
```

① 第 15 行至第 16 行定义菜单项 ID 常量。

② 第 26 行至第 53 行定义 onCreateOptionsMenu()方法。在其中为选项菜单添加两个菜单项,第 28 行至第 29 行添加菜单项 begmenuitem,其 ID 号为 ITEM1,其显示文本为“运动”,并为此菜单项设置了图标;同样,第 31 行至第 32 行添加菜单项 retmenuitem。第 35 行至第 48 行创建 OnMenuItemClickListener 监听对象 lsn,并定义回调方法 OnMenuItemClick()。第 50 行至第 51 行,为两个菜单项添加 OnMenuItemClickListener。

【运行结果】

在 Eclipse 中启动模拟器,然后运行项目 OptionMenuExample,按下模拟器的 MENU 键,显示选项菜单如图 5.27 所示;选择“运动”菜单项后的显示效果如图 5.28 所示。

5.3.2 子菜单

子菜单(SubMenu)是将相似功能的菜单项分组的一种菜单。

SubMenu 类位于 android.view 包下,继承自 Menu,每个 SubMenu 对象代表一个子菜单。

SubMenu 通常与 OptionMenu 联合使用,往菜单中添加子菜单使用 addSubMenu()方法。

SubMenu 常用方法如表 5.7 所示。



图 5.27　按下模拟器的 MENU 键，显示选项菜单



图 5.28　选择“运动”菜单项后的显示效果

表 5.7　SubMenu 常用方法

方　　法	说　　明
setHeaderIcon(icon/id)	使用 Drawable 对象或 id 资源设置 SubMenu 的标题图标
setIcon(icon/id)	使用 Drawable 对象或 id 资源设置在父菜单中显示的图标
setHeaderTitle(title/id)	使用标题文本对象或 id 资源设置 SubMenu 的标题
setHeaderView(View)	设置指定 View 对象作为子菜单的图标

【例 5.16】　子菜单与选项菜单联合应用举例。

【解题思路】

设置选项菜单，其中有两个菜单分别是“性别”子菜单和“爱好”子菜单。当接收用户选择了子菜单中的菜单项时，在屏幕的文本编辑框控件中累计记录所做的选择的内容。

【开发步骤和程序分析】

(1) 在 Eclipse 中创建一个 MenuExample 应用项目，包名为 com. application. menuexample。

(2) 准备图片资源，将图片资源复制到 res/drawable-mdpi 中。

(3) 准备字符串资源，在 res/values 目录下的 string. xml 文件中，编辑代码如下：

```
<?xml version = "1.0" encoding = "utf - 8"?>
< resources >
    < string name = "hello"> Hello World, MenuExample!</string >
```

```
    <string name = "app_name"> MenuExample </string>
    <string name = "label">请选择: \n </string>
    <string name = "gender">性别</string>
    <string name = "male">男</string>
    <string name = "female">女</string>
    <string name = "hobby">兴趣</string>
    <string name = "hobby0">阅读</string>
    <string name = "hobby1">运动</string>
    <string name = "hobby2">音乐</string>
    <string name = "hobby3">视频</string>
    <string name = "hobby4">绘画</string>
    <string name = "hobby5">摄影</string>
</resources>
```

(4) 设计布局，在 res/layout 目录下的 main.xml 文件中，定义垂直线性布局，在该布局中，设置一个 ScrollView 控件，其中设置一个 EditText 控件。在该文件中编辑代码如下：

```
1  <?xml version = "1.0" encoding = "utf - 8"?>
2  <! -- 定义一个垂直线性布局 LinearLayout -->
3  <LinearLayout
4       android:id = "@ + id/LinearLayout01"
5       android:background = "@drawable/background"
6       android:layout_width = "fill_parent"
7       android:layout_height = "fill_parent"
8       android:orientation = "vertical"
9       xmlns:android = "http://schemas.android.com/apk/res/android">
10      <! -- 设置一个 ScrollView 控件 -->
11      <ScrollView
12       android:id = "@ + id/ScrollView01"
13       android:layout_width = "fill_parent"
14       android:layout_height = "fill_parent">
15          <! -- 设置一个 EditText 控件 -->
16          <EditText
17              android:id = "@ + id/EditText01"
18              android:layout_width = "fill_parent"
19              android:layout_height = "fill_parent"
20              android:editable = "false"
21              android:cursorVisible = "false"
22              android:text = "@string/label">
23          </EditText>
24     </ScrollView>
25 </LinearLayout>
```

① 第 2 行至第 25 行定义一个垂直线性布局 LinearLayout。

② 第 11 行至第 24 行设置一个 ScrollView 控件，其中，第 16 行至第 23 行设置一个 EditText 控件，由于文本框不断添加文本，可能超过屏幕显示范围，所以将文本框放在滚动

视图中。

(5) 在 src/com. application. menuexamplem 包下的 MenuExampleActivity. java 文件中，定义 MenuExampleActivity 类继承自 Activity 类，重写菜单创建方法，分别定义“性别”子菜单和“兴趣”子菜单，并为子菜单添加菜单项，重写 onOptionsItemSelected()方法，该方法为单选或复选菜单项选中状态变化后的回调方法，定义 appendStateStr()方法，该方法为获取当前选择状态的方法。在该文件中编辑代码如下：

```
package com.application.menuexample;

import com.application.menuexample.R;
import android.app.Activity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.SubMenu;
import android.widget.EditText;

//定义 MenuExampleActivity 类继承自 Activity 类
public class MenuExampleActivity extends Activity {
    //定义菜单项 ID 常量
    final int MENU_GENDER_MALE = 0;
    final int MENU_GENDER_FEMALE = 1;
    final int MENU_HOBBY0 = 2;
    final int MENU_HOBBY1 = 3;
    final int MENU_HOBBY2 = 4;          /
    final int MENU_HOBBY3 = 5;
    final int MENU_HOBBY4 = 6;
    final int MENU_HOBBY5 = 7;
    final int MENU_GENDER = 8;
    final int MENU_HOBBY = 9;
    final int GENDER_GROUP = 0;
    final int HOBBY_GROUP = 1;
    final int MAIN_GROUP = 2;

    //定义一个 MenuItem[] 数组 myHobby,用于存储兴趣菜单项组
    MenuItem[] myHobby = new MenuItem[6];
    //定义一个菜单项变量 male,用于判断性别
    MenuItem male = null;

    @Override
    public void onCreate(Bundle savedInstanceState) {       //重写 onCreate 方法
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);                      //设置当前屏幕
  }
```

```
//重写菜单创建方法
@Override
public boolean onCreateOptionsMenu(Menu menu){
    //定义"性别"子菜单 subMenuGender
    SubMenu subMenuGender = menu.addSubMenu(
            MAIN_GROUP,MENU_GENDER,0,R.string.gender);
    subMenuGender.setIcon(R.drawable.sex);
    subMenuGender.setHeaderIcon(R.drawable.sex);
    male = subMenuGender.add(GENDER_GROUP, MENU_GENDER_MALE, 0, R.string.male);
    male.setChecked(true);
    subMenuGender.add(GENDER_GROUP, MENU_GENDER_FEMALE, 0, R.string.female);
    subMenuGender.setGroupCheckable(GENDER_GROUP, true,true);

    //定义"兴趣"子菜单 subMenuHobby
    SubMenu subMenuHobby = menu.addSubMenu(
    MAIN_GROUP,MENU_HOBBY,0,R.string.hobby);
    subMenuHobby.setIcon(R.drawable.reading);
    subMenuHobby.setHeaderIcon(R.drawable.reading);
    //为子菜单添加菜单项,且赋予菜单项数组 myHobby 中
    myHobby[0] = subMenuHobby.add(HOBBY_GROUP, MENU_HOBBY0, 0, R.string.hobby0);
    myHobby[1] = subMenuHobby.add(HOBBY_GROUP, MENU_HOBBY1, 0, R.string.hobby1);
    myHobby[2] = subMenuHobby.add(HOBBY_GROUP, MENU_HOBBY2, 0, R.string.hobby2);
    myHobby[3] = subMenuHobby.add(HOBBY_GROUP, MENU_HOBBY3, 0, R.string.obby3);
    myHobby[4] = subMenuHobby.add(HOBBY_GROUP, MENU_HOBBY4, 0, R.string.hobby4);
    myHobby[5] = subMenuHobby.add(HOBBY_GROUP, MENU_HOBBY5, 0, R.string.hobby5);
    myHobby[0].setCheckable(true);
    myHobby[1].setCheckable(true);
    myHobby[2].setCheckable(true);
    myHobby[3].setCheckable(true);
    myHobby[4].setCheckable(true);
    myHobby[5].setCheckable(true);
    return true;
}

//重写 onOptionsItemSelected()方法,该方法为单选或复选菜单项选中状态变化后的回调方法
@Override
public boolean onOptionsItemSelected(MenuItem mi){
    switch(mi.getItemId()){
        case MENU_GENDER_MALE:
        case MENU_GENDER_FEMALE:
            mi.setChecked(true);
            appendStateStr();
            break;
        case MENU_HOBBY0:
```

```
83                case MENU_HOBBY1:
84                case MENU_HOBBY2:
85                case MENU_HOBBY3:
86                case MENU_HOBBY4:
87                case MENU_HOBBY5:
88                    mi.setChecked(!mi.isChecked());
89                    appendStateStr();/
90                    break;
91            }
92            return true;
93        }
94
95        //定义 appendStateStr()方法,该方法为获取当前选择状态的方法
96        public void appendStateStr(){
97            String mychoice = "您选择的性别为: ";
98            if(male.isChecked()){
99                mychoice = mychoice + "男";
100            }
101            else{
102                mychoice = mychoice + "女";
103            }
104
105            String hobbyStr = "";
106            for(MenuItem mi:myHobby){
107                if(mi.isChecked()){
108                    hobbyStr = hobbyStr + mi.getTitle() + "、";
109                }
110            }
111            if(hobbyStr.length()> 0){
112                mychoice = mychoice + ",您的兴趣为: " + hobbyStr.substring(0, hobbyStr.length() - 1) +
113                        ".\n";
114            }
115            else{
116                mychoice = mychoice + ".\n";
117            }
118            EditText et = (EditText)MenuExampleActivity.this.findViewById(R.id.EditText01);
119            et.append(mychoice);
120        }
121 }
```

① 第 14 行至第 26 行定义菜单项 ID 常量,第 14 行至第 15 行定义“性别”子菜单项 ID 常量,第 16 行至第 21 行定义“兴趣”子菜单项 ID 常量,第 22 行定义“性别”子菜单编号,第 23 行定义“兴趣”子菜单编号,第 24 行定义性别子菜单项组的编号,第 25 行定义“兴趣”子菜单项组的编号,第 26 行定义外层总菜单项组的编号。

② 第 29 行定义一个 MenuItem[]数组 myHobby，用于存储"兴趣"菜单项组，第 31 行定义一个菜单项变量 male，用于判断"性别"，如果 male 选中则为男性，否则为女性。

③ 第 40 行至第 71 行重写菜单创建方法 onCreateOptionsMenu()。第 43 行至第 50 行为"性别"单选菜单创建实例，第 43 行至第 46 行定义"性别"子菜单 subMenuGender，第 47 行为子菜单添加一个菜单项。第 53 行至第 69 行为"兴趣"复选菜单创建实例，第 53 行至第 56 行定义"兴趣"子菜单 subMenuHobby，第 58 行至第 63 行为子菜单添加 6 个菜单项，且赋予菜单项数组 myHobby 中，第 64 行至第 69 行设置菜单项是可选的，即设置该子菜单为复选菜单，第 70 行返回定义好的菜单实例，完成 onCreateOptionsMenu(Menu menu)方法。

④ 第 74 行至第 93 行重写 onOptionsItemSelected()方法，该方法为单选或复选菜单项选中状态变化后的回调方法。

⑤ 第 96 行至第 120 行定义 appendStateStr()方法，该方法为获取当前选择状态的方法。

【运行结果】

在 Eclipse 中启动模拟器，然后运行项目 MenuExample，按下模拟器的 MENU 键，显示选项菜单如图 5.29 所示，选择"性别"菜单项后的显示效果如图 5.30 所示。



图 5.29　按下模拟器的 MENU 键，显示选项菜单



图 5.30　选择"性别"菜单项后的显示效果

选择"男"单选框项后的显示效果如图 5.31 所示，选择"兴趣"菜单项后的显示效果如图 5.32 所示。



图 5.31　选择“男”单选框项后的显示效果



图 5.32　选择“兴趣”菜单项后的显示效果

选择“阅读”复选框项后的显示效果如图 5.33 所示。



图 5.33　选择“阅读”复选框项后的显示效果

5.3.3 上下文菜单

当用户长时间按键不放时，弹出的菜单称为上下文菜单（ContentMenu）。

ContentMenu 位于 android. view 包下，继承自 Menu。

ContentMenu 注册于某个 View 对象上，当长按下该 View 对象时，呼出上下文菜单。

ContentMenu 菜单项不支持快捷键，不附带图标。ContentMenu 标题可以指定图标。

ContentMenu 常用方法如表 5.8 所示。

表 5.8 ContentMenu 常用方法

方　　法	说　　明
onCreateContextMenu()	每次为 View 对象呼出上下文菜单时都调用
onContextItemSelected()	当用户选择了上下文菜单选项后调用
onContextMenuClosed()	当上下文菜单被关闭时调用
registerForContextMenu()	为指定的 View 对象注册一个上下文菜单

在程序中创建上下文菜单的步骤如下：

（1）重写 Activity 的 onCreateContextMenu()方法，调用 Menu 的 add()方法添加菜单项 MenuItem。

（2）重写 Activity 的 onContextItemSelected()方法，响应菜单单击事件。

（3）调用 registerForContextMenu()方法，为视图 View 对象注册上下文菜单。

5.4 小　　结

本章主要介绍了以下内容：

（1）Android 事件的处理机制有两种：一种是基于监听接口的事件处理，一种是基于回调机制的事件处理。

在基于监听接口的事件处理模型中，有三个对象：Event Source（事件源）、Event（事件）、Event Listener（事件监听器）。事件监听处理流程为：将事件监听器注册到事件源，触发事件源上的事件，生成事件对象，事件被发送到事件监听器，调用事件监听器中相应的事件处理方法来处理事件并给出响应。

基于回调机制的事件处理是将事件的处理绑定在控件上，由用户界面控件自己处理事件，回调机制需要自定义 View 来实现。每个 View 类都有自己的处理事件的回调方法，开发人员可以通过重写 View 中这些回调方法来实现需要响应的事件。

（2）在 Android 中，Widget 常用高级控件有：与适配器相关的控件、其他与视图相关的控件、进度条与滑块进度条等。

（3）与适配器相关的控件有 AutoCompleteTextView（自动完成文本框）、Spinner（下拉列表）、Gallery（画廊视图）、ListView（列表视图）、GridView（网格视图）。

AutoCompleteTextView 在用户输入一定的字符串后，会弹出一个下拉菜单，供用户选择，用户选择某个菜单后，它会按用户选择自动填写该文本框。

Spinner 是一个垂直下拉列表框，只有当用户单击这个控件时，才会下拉出选项列表供用户选择。

Gallery 是水平滚动显示图片资源的列表。

ListView 以垂直的、可滚动的列表方式显示一组列表项的视图。

GridView 是一种以二维表格形式显示控件的视图。

(4) 其他与视图相关的控件有 ScrollView（滚动视图）、TabHost（选项卡）、ImageSwitcher（图像切换器）。

ScrollView 控件的内容在一屏显示不完时，便会自动产生纵向滚动功能，以显示被挡住的部分内容。

TabHost 中标签内容对应于页面内容，通过一页中多个标签，可在一页中显示多个页面内容。

ImageSwitcher 控件用于多张图片的切换。

(5) 进度条与滑块进度条有 ProgressBar（进度条）、SeekBar（拖动条）、RatingBar（星级评分条）。

ProgressBar（进度条）是一种实用控件，通常用于加载资源或执行某些耗时操作。

SeekBar（拖动条）是一种滑块控件，继承自 ProgressBar，用来接收用户拖拉滑块操作，通常用于调节声音大小等操作。

RatingBar（星级评分条）是另一种滑块控件，外观是 5 个星星，通常用于星级评分等操作。

(6) 菜单（Menu）设计在人机交互中是十分重要的，它提供了不同功能分组展示能力，应用十分广泛。

Android 平台下有三类菜单：选项菜单（OptionMenu）、子菜单（Submenu）、上下文菜单（ContentMenu）。

选项菜单（OptionMenu）定义在 android. view. Menu 包中。一个选项菜单是一个 Menu 对象，在 Menu 对象中可以添加菜单项 MenuItem。

子菜单（SubMenu）是将相似功能的菜单项分组的一种菜单。SubMenu 类位于 android. view 包下，它继承自 Menu，每个 SubMenu 对象代表一个子菜单。

上下文菜单（ContentMenu）位于 android. view 包下，继承自 Menu。

习　题　5

一、选择题

5.1　基于回调机制的事件处理需要自定义________来实现。

A. Event　　B. View

C. Event Listener　　D. Event Sourcer

5.2　下面哪一个选项不是基于监听接口的事件处理模型中的对象？________

A. Event　　B. Event Source　　C. Bundle　　D. Event Listener

5.3　水平滚动显示图片资源列表的高级控件是________。

A. ListView　　B. GridView　　C. Gallery　　D. Spinner

5.4　以二维表格形式显示视图的高级控件是________。

A. ListView　　B. GridView　　C. Gallery　　D. Spinner

5.5　通过一页中多个标签来显示多个页面内容的高级控件是________。

A. ScrollView　　B. AutoCompleteTextView

C. ImageSwitcher　　D. TabHost

5.6　当一屏显示不完时,便会自动产生纵向滚动功能以显示被挡住部分的高级控件是________。

A. ScrollView　　B. AutoCompleteTextView

C. ImageSwitcher　　D. TabHost

5.7　进度条控件是________。

A. SeekBar　　B. ListView　　C. RatingBar　　D. ProgressBar

5.8　用于星级评分等操作的滑块控件是________。

A. SeekBar　　B. ListView　　C. RatingBar　　D. ProgressBar

二、填空题

5.9　基于监听接口的事件处理模型的三个对象是:Event Source、Event 和________。

5.10　Spinner 是一个________列表框。

5.11　ListView 以________可滚动的列表方式显示一组列表项的视图。

5.12　ImageSwitcher 用于多张图片的________。

5.13　ImageSwitcher 控件常与________控件一起使用。

5.14　SeekBar 是一种滑块控件,继承自________。

5.15　一个选项菜单是一个________对象。

5.16　子菜单是将________的菜单项分组的一种菜单。

5.17　SubMenu 通常与________联合使用。

三、问答题

5.18　简述事件监听处理流程。

5.19　Adapter 有何作用? 它可分为哪几种?

5.20　使用 ArrayAdapter 为下拉列表加载数据,有哪两种方式?

5.21　试比较 Gallery 控件和 Spinner 控件。

5.22　试比较 GridView 控件和 ListView 控件。

5.23　简述 Android 三类菜单的特点。

四、应用题

5.24　以 Spinner 方式设计一个应用项目,用于显示省市列表,当用户选择下拉列表中的某省后,立即列出该省的城市名称供用户选择。

5.25　设计一个动态下拉列表,当用户在 EditText 中写入文本,单击"添加"按钮即将文本存储在 Spinner 中;当用户在下拉列表中单击某项,该选项的文本显示在 EditText 中;选择下拉列表中某项,单击"删除"按钮,该选项被删除。

5.26　设计一个应用项目,有两个控件 TextView 和 EditText,在 EditText 中输入

1～100 的分数，单击“转换”按钮后，按照输入的分数段：90～100、80～89、70～79、60～69、0～59，在 TextView 中分别显示“优秀”“良好”“中等”“及格”“不及格”，如果输入不合要求，则弹出“输入错误”提示，要求重新输入。

5.27 使用 SubMenu 和 OptionMenu 设计一个应用项目界面，在选项菜单中包括“性别”子菜单，“文化程度”子菜单。

5.28 设计一个网上购书应用项目的界面，有“新书展示”“人文社科”“文学”“生活”“科技”等模块。

5.29 设计一个网上购手机的应用项目的界面，有“品牌”“价格”“网络”“热点”“屏幕尺寸”等模块。

5.30 设计一个个人主页的应用项目的界面，有“主页”“日志”“相册”“留言板”“音乐”等模块。

第6章 后台服务

本章要点

- Service 组件适用于没有用户界面，并长时间在后台运行的应用。
- Service 的启动方式有两种：通过 startService 启动和通过 bindService 启动。
- BroadcastReceiver 是对广播消息进行过滤并响应的组件，不包含任何用户界面，其监听的事件源是其他组件。
- 在 Android 消息提示机制中，Toast 是一种快速的即时消息，消息内容简短，悬浮于应用程序的最上方，而 Notification 消息内容显示于手机的状态条中。

在 Android 应用中，有一些应用是没有用户界面，而且可以在运行其他应用时同时运行，例如播放音乐、接收和发送广播消息等。本章介绍 Service 组件及其生命周期、Service 的启动模式和绑定模式、BroadcastReceiver 组件、Notification 等内容。

6.1 Service 组件及其生命周期

Service 是 Android 系统的服务组件，用于没有用户界面，又需要长时间在后台运行的应用。

6.1.1 Service 简介

由于手机硬件性能和屏幕尺寸的限制，通常 Android 系统仅允许一个应用程序处于激活状态并显示在手机屏幕上，而暂停其他处于未激活状态的程序。因此，Android 系统需要一种后台服务机制，允许在没有用户界面的情况下，使程序能够长时间在后台运行，能够进行事件处理或数据更新。

Service 组件适用于没有用户界面，并长时间在后台运行的应用，例如播放音乐、检测 SD 卡上文件的变化、后台数据计算和发送通知等。

Service 与 Activity 是两个相似的组件，它们都代表可执行的程序。其区别为：Service 是长时间在后台运行、没有用户界面的 Android 组件，Activity 的作用是提供用户界面并与用户交互等。

Service 可分为本地服务（Local Service）和远程服务（Remote Service）两类。本地服务用于应用程序内部，远程服务用于 Android 系统内部的应用程序之间。

6.1.2 Service 的生命周期

Service 有自己的生命周期。Service 的生命周期方法比 Activity 要少一些，只有

onCreate()、onStart()和 onDestroy()等方法。

- onCreate()方法

当 Service 第一次被创建时，由系统调用。

- onStart(Intent intent, int startId)方法

当 startService 方法启动 Service 时，该方法被调用。

- onDestroy()

当 Service 不再使用时，由系统调用。

一个 Service 只会创建一次，销毁一次，但可以开始多次，因此，onCreate()和 onDestroy()方法只会被调用一次，而 onStart()方法会被调用多次。

Service 没有可与用户交互的用户界面，它不能自己启动，需要通过某一个 Activity 或者其他 Context 对象来启动。Service 的启动方式有两种：通过 startService 启动和通过 bindService 启动。

1. 通过 startService 启动

通过 startService 启动 Service，依次调用 onCreate→onStart(或 onStartCommand())方法。

如果 Service 已经启动了，再次启动 Service 时，不会再执行 onCreate()方法，而是直接执行 onStart 方法。

当 Service 停止时直接进入 onDestroy。无论之前调用了几次 startService，只要调用一次 stopService，将结束该 Service。

通过 startService 启动的 Service 生命周期如图 6.1(a)所示。

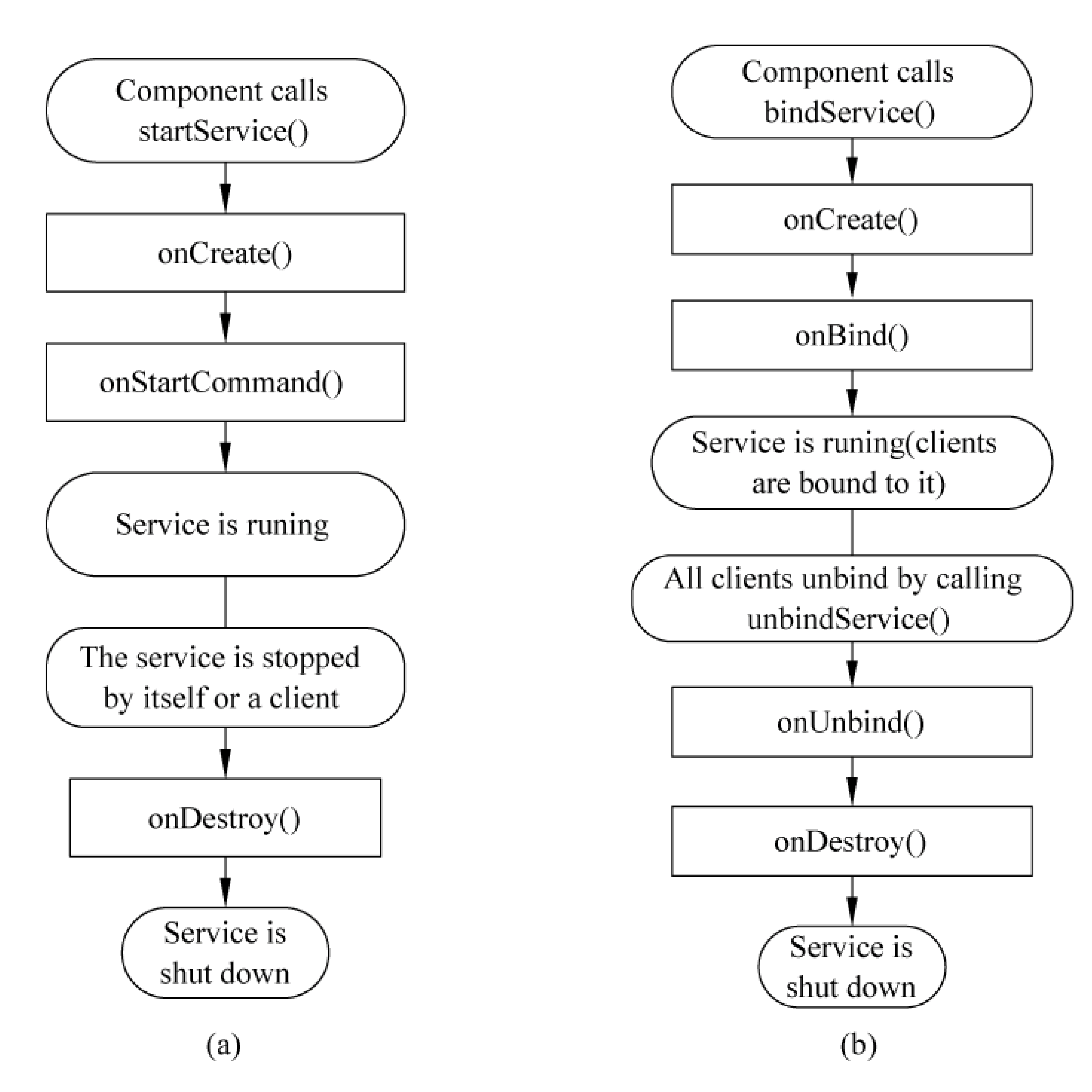


图 6.1 通过 startService 启动的 Service 生命周期与通过 bindService 启动的 Service 生命周期

2. 通过 bindService 启动

通过 bindService 启动 Service，运行 onCreate→onBind，此时，将调用者和 Service 绑定在一起。

当调用者退出了 Service，Service 就会调用 onUnbind→onDestroy，与调用者同时退出，这就是绑定的含义。

通过 bindService 启动的 Service 生命周期如图 6.1(b)所示。

6.2 Service 的启动模式和绑定模式

Service 一般由 Activity 启动，也可由其他 Service 或者 BroadcastReceiver 启动。

(1) 创建 Service 子类需要重写的方法使用 Service 进行编程，首先要定义一个子类继承 Service 类，并重写该类中相应的方法，这些方法如下：

- onCreate()

当 Service 第一次被创建时，由系统调用。

- onStart(Intent intent, int startId)

当 startService()方法启动 Service 时，该方法被调用。

- onBind(Intent intent)

该方法返回一个绑定的接口给 Service。

- onDestroy()

当 Service 不再使用时，由系统调用。

(2) 启动 Service：定义好一个 Service 后，就可以在其他组件中启动该 Service，启动一个 Service 与启动一个 Activity 很相似，即传递一个 Intent。

(3) 注册 Service 组件：在应用程序中使用 Service，需要在 AndroiManifest.xml 文件中显式地注册< service >标签。

在 AndroidManifest.xml 文件中注册 Service 的代码如下：

```
<service android:name = ".MyService"/>
```

6.2.1 启动模式下的 Service

在启动模式下，有两种方法启动 Service：显式启动和隐式启动。

1. 显式启动

1) 启动 Service

显式启动通过类名称来启动，需要在 Intent 中指明 Service 所在的类，并调用 startService(Intent)启动 Service，显式启动代码如下：

```
final Intent serviceIntent = new Intent(this, MyService.class);
startService(serviceIntent);
```

在上面的代码中，Intent 指明了启动的 Service 所在类为 MyService。

2）停止 Service

显式启动停止 Service,需要将启动 Service 的 Intent 传递给 stopService(Intent)函数,代码如下：

```
stopService(serviceIntent);
```

2. 隐式启动

1）启动 Service

隐式启动通过 Intent Filter 来启动 Service,需要设置 Intent 的 action 属性,这样可以在不声明 Service 所在类的情况下启动服务,隐式启动的代码如下：

```
final Intent serviceIntent = new Intent();
serviceIntent.setAction("com.application.MyService");
```

2）停止 Service

隐式启动停止 Service 的代码与显式启动停止 Service 的代码相同。

如果 Service 和调用服务的组件在同一个应用程序中,可以使用显式启动或隐式启动,显式启动更加易于使用,且代码简洁。但如果服务和调用服务的组件在不同的应用程序中,则只能使用隐式启动。

下面举例说明 Service 显式启动并介绍 Toast。

Toast 是一种快速的即时消息,消息内容简短,悬浮于应用程序的最上方,不获得焦点。

Toast 类在 android. widget 包下,Toast 对象的创建通过 makeText()方法实现,Toast 对象通过 show()方法显示在屏幕上,Toast 对象一般用于某项操作执行后是否成功的消息提示。

makeText()方法格式如下：

```
makeText(Context context, String message, int duration)
```

其中,第 1 个参数是 Toast 对象的当前上下文引用,第 2 个参数是一个字符串,内容为当前显示的消息,第 3 个参数是 Toast 显示的时间长度。

使用 Toast 显示消息"调用成功!"的代码举例如下：

```
Toast.makeText(this, "调用成功!", Toast.LENGTH_LONG).show();
```

【例 6.1】 Service 显式启动举例。

【解题思路】

定义一个子类继承 Service 类,并重写该类中相应的方法 onCreate()、onStart()和 onDestroy(),在上述 3 个方法中,分别使用 Toast 显示文本："(1)调用 onCreate()""(2)调用 onStart()""(3)调用 onDestroy()"。

定义一个 Activity,其中的两个按钮分别用于启动 Service 和销毁 Service。

【开发步骤和程序分析】

(1) 在 Eclipse 中创建一个 ExplicitStartService 应用项目,包名为 com. application. explicitstartservice。

(2) 设计布局,在 res/layout 目录下的 main. xml 文件中,定义垂直线性布局 LinearLayout,在该布局中,设置一个 TextView 控件,两个 Button 控件。

在该文件中编辑代码如下：

```
1  <?xml version = "1.0" encoding = "utf - 8"?>
2  <! -- 定义一个垂直线性布局 -->
3  < LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
4       android:orientation = "vertical"
5       android:layout_width = "fill_parent"
6       android:layout_height = "fill_parent"
7  >
8       <! -- 设置一个 TextView 控件 -->
9       < TextView   android:id = "@ + id/label"
10        android:layout_width = "fill_parent"
11        android:layout_height = "wrap_content"
12        android:text = "@string/hello">
13      </TextView >
14      <! -- 设置一个 Button 控件,按钮名为"启动后台服务" -->
15      < Button android:id = "@ + id/start"
16          android:layout_width = "wrap_content"
17          android:layout_height = "wrap_content"
18          android:text = "启动后台服务" >
19      </Button >
20      <! -- 设置一个 Button 控件,按钮名为"停止后台服务" -->
21     < Button android:id = "@ + id/stop"
22          android:layout_width = "wrap_content"
23          android:layout_height = "wrap_content"
24          android:text = "停止后台服务" >
25     </Button >
26 </LinearLayout >
```

① 第 2 行至第 22 行定义一个垂直线性布局 LinearLayout。

② 第 7 行至第 11 行设置一个 TextView 控件。

③ 第 12 行至第 16 行、第 17 行至第 21 行分别设置两个按钮控件，按钮名分别为“启动后台服务”和“停止后台服务”。

（3）在 com.application.optionmenuexample 包下的 ExplicitService.java 文件中，定义 ExplicitService 类继承自 Service 类，重写 onCreate()方法、onStart ()方法、onDestroy()方法，在上述方法中，使用 Toast 显示调用消息。在该文件中编辑代码如下：

```
package com.application.explicitstartservice;

import android.app.Service;
import android.content.Intent;
import android.os.IBinder;
import android.widget.Toast;

//定义 ExplicitService 类继承自 Service 类
```

```
9   public class ExplicitService extends Service{
10
11      //重写 onCreate()方法
12      @Override
13      public void onCreate() {
14          super.onCreate();
15          //使用 Toast 显示消息:"(1) 调用 onCreate()"
16          Toast.makeText(this, "(1) 调用 onCreate()", Toast.LENGTH_LONG).show();
17      }
18
19      //重写 onStart ()方法
20      @Override
21      public void onStart(Intent intent, int startId) {
22            super.onStart(intent, startId);
23          //使用 Toast 显示消息:"(2) 调用 onStart()"
24            Toast.makeText(this, "(2) 调用 onStart()", Toast.LENGTH_SHORT).show();
25      }
26
27      //重写 onDestroy()方法
28      @Override
29      public void onDestroy() {
30           super.onDestroy();
31          //使用 Toast 显示消息:"(3) 调用 onDestroy()"
32           Toast.makeText(this, "(3) 调用 onDestroy()", Toast.LENGTH_SHORT).show();
33      }
34
35      @Override
36      public IBinder onBind(Intent intent) {
37          return null;
38      }
39
40  }
```

① 第 9 行至第 40 行定义一个继承 Service 类的子类 ExplicitService。

② 第 12 行至第 17 行重写 onCreate()方法，在该方法中，第 16 行使用 Toast 显示消息："(1)调用 onCreate()"。

③ 第 20 行至第 25 行重写 onStart ()方法，在该方法中，第 24 行使用 Toast 显示消息："(2)调用 onStart()"。

④ 第 28 行至第 33 行重写 onDestroy()方法，在该方法中，第 32 行使用 Toast 显示消息："(3)调用 onDestroy()"。

(4) 在 com. application. optionmenuexample 包下的 ExplicitStartServiceActivity. java 文件中，定义 ExplicitStartServiceActivity 类继承自 Activity 类，创建两个按钮对象分别用于启动 Service 和销毁 Service。在该文件中编辑代码如下：

```
1  package com.application.explicitstartservice;
2
3  import com.application.explicitstartservice.R;
4  import android.app.Activity;
5  import android.content.Intent;
6  import android.os.Bundle;
7  import android.view.View;
8  import android.widget.Button;
9
10 //定义 ExplicitStartServiceActivity 类继承自 Activity 类
11 public class ExplicitStartServiceActivity extends Activity {
12
13     @Override
14     public void onCreate(Bundle savedInstanceState) {
15         super.onCreate(savedInstanceState);
16         setContentView(R.layout.main);
17
18         //创建按钮对象
19         Button startButton = (Button)findViewById(R.id.start);
20         Button stopButton = (Button)findViewById(R.id.stop);
21         //创建 Intent 对象 serviceIntent
22         final Intent serviceIntent = new Intent(this, ExplicitService.class);
23
24         startButton.setOnClickListener(new Button.OnClickListener(){
25             public void onClick(View view){
26                 //通过 startService()方法以 serviceIntent 为参数启动 Service
27                 startService(serviceIntent);
28             }
29         });
30
31         stopButton.setOnClickListener(new Button.OnClickListener(){
32             public void onClick(View view){
33                 //通过 stopService()方法以 serviceIntent 为参数销毁 Service
34                 stopService(serviceIntent);
35             }
36         });
37     }
38 }
```

① 第 19 行创建按钮对象 startButton，第 20 行创建按钮对象 stopButton。

② 第 22 行创建 Intent 对象 serviceIntent。

③ 第 27 行是启动 Service 的代码，通过 startService()方法以 serviceIntent 为参数启动 Service。

④ 第 34 行是销毁 Service 的代码，通过 stopService()方法以 serviceIntent 为参数销毁 Service。

(5) 添加 Service 组件声明，在 AndroidManifest.xml 文件中声明一个 Service 组件，其代码如下：

```
1 <?xml version = "1.0" encoding = "utf - 8"?>
2 <manifest xmlns:android = "http://schemas.android.com/apk/res/android"
3       package = "com.application.explicitstartservice"
4       android:versionCode = "1"
5       android:versionName = "1.0" >
6       <uses - sdk android:minSdkVersion = "14" />
7       <application android:icon = "@drawable/ic_launcher"
8           android:label = "@string/app_name" >
9           <activity android:label = "@string/app_name"android:name
10              = "com.application.explicitstartservice.ExplicitStartServiceActivity" >
11              <intent - filter >
12                  <action android:name = "android.intent.action.MAIN" />
13                  <category android:name = "android.intent.category.LAUNCHER" />
14              </intent - filter >
15          </activity >
16          <service android:name = "com.application.explicitstartservice.ExplicitService"/>
17      </application >
18 </manifest >
```

第 16 行在< service >标签中，声明 ExplicitService 所在的 Service 类。

【运行结果】

在 Eclipse 中启动模拟器，然后运行项目 ExplicitStartService，初始界面如图 6.2 所示。单击“启动后台服务”按钮后，首先显示消息“(1)调用 onCreate()”，如图 6.3 所示。



图 6.2　Service 显式启动初始界面



图 6.3　单击“启动后台服务”按钮后首先显示消息“(1)调用 onCreate()”

然后显式消息“(2)调用 onStart()”，如图 6.4 所示。单击“停止后台服务”按钮后显示消息“(3)调用 onDestroy()”，如图 6.5 所示。



图 6.4　然后显示消息“(2)调用 onStart()”



图 6.5　单击“停止后台服务“按钮后显示消息“(3)调用 onDestroy()”

6.2.2　绑定模式下的 Service

本节介绍使用 bindService()方法启动 Service 和使用 unbindService()方法取消绑定。

1. 使用 bindService()方法启动 Service

绑定模式使用 bindService()方法启动 Service，其格式如下：

```
bindService(Intent service, ServiceConnection conn,int flags);
```

其中的参数说明如下：

(1) service：该参数通过 Intent 指定需要启动的 service。

(2) conn：该参数是 ServiceConnection 对象，当绑定成功后，系统将调用 ServiceConnnection 的 onServiceConnected()方法，当绑定意外断开后，系统将调用 ServiceConnnection 中的 onServiceDisconnected 方法。

(3) flags：该参数指定绑定时是否自动创建 Service。如果指定为 BIND_AUTO_CREATE，则自动创建，指定为 0，则不自动创建。

绑定方式中，当调用者通过 bindService()函数绑定 Service 时，onCreate()函数和 onBinde()函数将被先后调用。

2. 使用 unbindService()方法取消绑定

取消绑定仅需要使用 unbindService()方法，并将 ServiceConnnection 传递给

unbindService()方法。

但需要注意的是，unbindService()方法成功后，系统并不会调用 onServiceConnected()，因为 onServiceConnected()仅在意外断开绑定时才被调用。

当调用者通过 unbindService()函数取消绑定 Service 时，onUnbind()函数将被调用。如果 onUnbind()函数返回 True，则表示重新绑定服务时，onRebind()函数将被调用。

【例 6.2】 使用绑定方式启动 Service 举例。

【解题思路】

为了使 Service 支持绑定，定义一个子类继承 Service 类，并在 Service 类中重写 onBind()方法，在 onBind()方法中返回 Service 实例。

定义一个 Activity，其中的三个按钮分别用于绑定后台服务、取消绑定和乘法运算。

【开发步骤和程序分析】

(1) 在 Eclipse 中创建一个 BindService 应用项目，包名为 com. application. bindservice。

(2) 设计布局，在 res/layout 目录下的 main. xml 文件中，定义垂直线性布局 LinearLayout，在该布局中，设置一个 TextView 控件，三个 Button 控件。在该文件中编辑代码如下：

```
<?xml version = "1.0" encoding = "utf - 8"?>
<! -- 定义一个垂直线性布局 -->
LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = "vertical"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
>
    <! -- 设置一个 TextView 控件 -->
    < TextView  android:id = "@ + id/label"
        android:layout_width = "fill_parent"
        android:layout_height = "wrap_content"
        android:text = "@string/hello">
    </TextView >
    <! -- 设置一个 Button 控件,按钮名为"绑定后台服务" -->
    < Button android:id = "@ + id/bind"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:text = "绑定后台服务" >
    </Button >
    <! -- 设置一个 Button 控件,按钮名为"取消绑定" -->
    < Button android:id = "@ + id/unbind"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:text = "取消绑定" >
    </Button >
    <! -- 设置一个 Button 控件,按钮名为"乘法运算" -->
    < Button android:id = "@ + id/multiplication"
```

```
28            android:layout_width = "wrap_content"
29            android:layout_height = "wrap_content"
30            android:text = "乘法运算" >
31        </Button>
32 </LinearLayout>
```

① 第 2 行至第 32 行定义一个垂直线性布局 LinearLayout。

② 第 9 行至第 13 行设置一个 TextView 控件。

③ 第 15 行至第 19 行、第 21 行至第 25 行、第 27 行至第 31 行分别设置三个按钮控件，按钮名分别为“绑定后台服务”“取消绑定”和“乘法运算”。

(3) 在 com. application. bindservice 包下的 MyService. java 文件中，定义 MyService 类继承自 Service 类，重写 onBind()方法、onUnbin()方法，在上述方法中，使用 Toast 显示调用消息，定义 Multiplication ()方法，用于乘法计算。在该文件中编辑代码如下：

```
package com.application.bindservice;

import android.app.Service;
import android.content.Intent;
import android.os.Binder;
import android.os.IBinder;
import android.widget.Toast;

//定义 MyService 类继承自 Service 类
public class MyService extends Service{

    //声明 IBinder 接口的一个接口变量 mBinder
    private final IBinder mBinder  =  new LocalBinder();

    //LocalBinder 是继承 Binder 的一个内部类
    public class LocalBinder extends Binder{
        MyService getService() {
            return MyService.this;
         }
    }

    //重写 onBind()方法,在该方法中,使用 Toast 显示消息: "本地绑定: MyService"
    @Override
    public IBinder onBind(Intent intent) {
        Toast.makeText(this, "本地绑定: MyService", Toast.LENGTH_SHORT).show();
        return mBinder;
    }

    //重写 onUnbin()方法,在该方法中,使用 Toast 显示消息: "取消本地绑定: MyService"
    @Override
    public boolean onUnbind(Intent intent){
```

```
32          Toast.makeText(this, "取消本地绑定：MyService", Toast.LENGTH_SHORT).show();
33          return false;
34      }
35
36      //定义 Multiplication ()方法,用于乘法计算
37      public long Multiplication(long a, long b){
38          return a * b;
39      }
40  }
```

① 第 10 行至第 40 行定义一个继承 Service 类的子类 MyService。

② 第 13 行，声明 IBinder 接口的一个接口变量 mBinder，IBinder 是用于进程内部和进程间过程调用的轻量级接口，mBinder 符合 onBind()方法返回值的要求，可将 mBinder 传递给调用者。

③ 第 16 行至第 20 行，LocalBinder 是继承 Binder 的一个内部类，其中，第 17 行至第 19 行实现了 getService()函数，当调用者获取到 mBinder 后，通过调用 getService()即可获取到 Service 实例。

④ 第 23 行至第 27 行重写 onBind()方法，在该方法中，第 25 行使用 Toast 显示消息"本地绑定：MyService"，第 26 行返回 mBinder。

⑤ 第 30 行至第 34 行重写 onUnbin()方法，在该方法中，第 32 行使用 Toast 显示消息"取消本地绑定：MyService"。

⑥ 第 37 行至第 39 行定义 Multiplication ()方法，用于乘法计算。

（4）在 com.application.bindservice 包下的 BindServiceActivity.java 文件中，定义 BindServiceActivity 类继承自 Activity 类，创建按钮对象 bindButton、unbindButton 和 multiplicationButton，分别为按钮对象 bindButton、unbindButton 和 multiplicationButton 绑定监听器。在该文件中编辑代码如下：

```
package com.application.bindservice;

import com.application.bindservice.R;
import android.app.Activity;
import android.content.ComponentName;
import android.content.Context;
import android.content.Intent;
import android.content.ServiceConnection;
import android.os.Bundle;
import android.os.IBinder;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;

//定义 BindServiceActivity 类继承自 Activity 类
public class BindServiceActivity extends Activity {
```

```
    private MyService myService;
    private boolean isBound = false;
    TextView labelView;

    //重写 onCreate()方法
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        labelView = (TextView)findViewById(R.id.label);
        //以下三行,分别创建按钮对象 bindButton、unbindButton 和 multiplicationButton
        Button bindButton = (Button)findViewById(R.id.bind);
        Button unbindButton = (Button)findViewById(R.id.unbind);
        Button multiplicationButton = (Button)findViewById(R.id.multiplicatione);

        //为 bindButton 按钮的单击事件绑定监听器
        bindButton.setOnClickListener(new View.OnClickListener(){

             @Override
             public void onClick(View v) {
                if(!isBound){
                final Intent serviceIntent = new Intent(BindServiceActivity.this,MyService.class);
                bindService(serviceIntent,mConnection,Context.BIND_AUTO_CREATE);
                isBound = true;
                }
            }
        });

        //为 unbindButton 按钮的单击事件绑定监听器
        unbindButton.setOnClickListener(new View.OnClickListener(){

            @Override
            public void onClick(View v) {
                if(isBound){
                    isBound = false;
                    unbindService(mConnection);
                    myService = null;
                }
            }
        });

        //为 multiplicationButton 按钮的单击事件绑定监听器
        multiplicationButton.setOnClickListener(new View.OnClickListener(){

            @Override
```

```
62              public void onClick(View v) {
63                  if (myService == null){
64                      labelView.setText("未绑定服务");
65                      return;
66                  }
67                  long a = Math.round(Math.random() * 100);
68                  long b = Math.round(Math.random() * 100);
69                  long result = myService.Multiplication(a, b);
70                  String msg = String.valueOf(a) + " * " + String.valueOf(b) +
71                      " = " + String.valueOf(result);
72                  labelView.setText(msg);
73              }
74          });
75      }
76
77      private ServiceConnection mConnection = new ServiceConnection() {
78
79          @Override
80          public void onServiceConnected(ComponentName name, IBinder service) {
81              myService = ((MyService.LocalBinder)service).getService();
82          }
83
84          @Override
85          public void onServiceDisconnected(ComponentName name) {
86              myService = null;
87          }
88      };
89 }
```

① 第 28 行至第 30 行，分别创建按钮对象 bindButton、unbindButton 和 multiplicationButton。

② 第 33 行至第 43 行，为 bindButton 按钮的单击事件绑定监听器，当用户单击 bindButton 按钮时，程序触发 OnClickListener 监听器，该监听器包含的事件处理方法通过 bindService()方法绑定服务。

③ 第 46 行至第 56 行，为 unbindButton 按钮的单击事件绑定监听器，当用户单击该按钮时，程序触发 OnClickListener 监听器，其包含的事件处理方法通过 unbindService()方法取消绑定。

④ 第 59 行至第 75 行，为 multiplicationButton 按钮的单击事件绑定监听器，当用户单击该按钮时，程序触发 OnClickListener 监听器，如果未绑定服务时单击“乘法运算”按钮，则显示“未绑定服务”，如果绑定服务时单击“乘法运算”按钮，则显示乘法算式。

【运行结果】

在 Eclipse 中启动模拟器，然后运行项目 BindService，初始界面如图 6.6 所示，单击“绑定后台服务”按钮后显示效果如图 6.7 所示。



图 6.6　Service 绑定方式启动初始界面



图 6.7　单击“绑定后台服务”按钮后显示“本地绑定：MyService”

单击“乘法运算”按钮后显示乘法算式如图 6.8 所示，单击“取消绑定”按钮后显示效果如图 6.9 所示。



图 6.8　单击“乘法运算”按钮后显示乘法算式



图 6.9　单击“取消绑定”按钮后显示“取消本地绑定：MyService”

6.2.3 线程使用

线程(Thread)是比进程更小的执行单位,多个线程可以并行工作。

1. 主线程和子线程

当一个程序首次启动时,Android 会启动一个 Linux 进程和一个主线程。主线程负责处理与 UI 相关的事件,并把相关的事件分发到对应的组件进行处理,所以主线程通常又叫作 UI 线程。Android UI 操作必须在 UI 线程中执行,Android 的 UI 是单线程(Single-threaded)。

子线程是非 UI 线程,子线程一般都是后台线程,在宏观上可以看作子线程是独立于主线程的,且能与主线程并行工作的程序单元。

在 Android 系统中,Activity、Service 和 BroadcastReceiver 都是工作在主线程上,任何耗时的处理过程都会降低用户界面的响应速度,甚至导致用户界面失去响应。"耗时的处理过程"一般指复杂运算过程、大量的文件操作、存在延时的网络通信和数据库操作等。

较好的解决方法是将耗时的处理过程转移到子线程上,这样可以缩短主线程的事件处理时间,避免用户界面响应速度过慢,从而失去响应。

2. 线程的创建和终止

创建线程步骤如下:

(1) 实现 Runnable 接口,并重写 run()方法,在 run()方法中放置代码的主体部分。

(2) 创建 Thread 对象,并将 Runnable 对象作为参数传递给 Thread 对象。

(3) 调用 start()方法启动线程。

当线程在 run()方法返回后,线程就自动终止了,也可以调用 stop()在外部终止线程,但这种方法并不安全,可能会产生异常。

最好的方法是通知线程自行终止,一般调用 interrupt()方法通知线程准备终止,代码中需要捕获 InterruptedException 异常,保证安全终止线程。

3. Handler 的使用

Handler 类位于 android.os 包下,Handler 允许将 Runnable 对象发送到线程的消息队列中,每个 Handler 实例绑定到一个单独的线程和消息队列上。当用户建立一个新的 Handler 实例,通过 post()方法将 Runnable 对象从后台线程发送给 GUI 线程的消息队列,当 Runnable 对象通过消息队列后,这个 Runnable 对象将被运行。

Android 的子线程不能直接操作 UI。如果需要可以尝试使用 Handler 消息传递机制。Handler 消息传递机制为:一个 Handler 对应一个 Activity,自定义的后台线程可与 Handler 通信,Handler 将与 UI 线程一起工作。

开发 Handler 类的步骤如下:

(1) 在 Activity 或 Activity 的 Widget 中开发 Handler 类的对象,重写 handlerMessage 方法。

(2) 在新启动的线程中调用 sendEmptyMessage 或 sendMessage 方法向 Handler 发送消息。

(3) Handler 类的对象用 handlerMessage 方法接收消息,然后根据消息执行相应的操作。

【例 6.3】 使用线程产生随机数,当单击"启动后台服务"按钮时,将启动后台线程,后台线程每秒产生一个随机数,并通过 Handler 将产生的随机数显示在界面上。当单击"停止后台服务"按钮时,将关闭后台线程。

【解题思路】

MyService.java 文件是定义 Service 的文件，用来创建线程、产生随机数和调用界面更新方法。

ThreadServiceActivity.java 文件是用户界面的 Activity 文件，封装 Handler 界面更新方法就在该文件中。

【开发步骤和程序分析】

(1) 在 Eclipse 中创建一个 ThreadService 应用项目，包名为 com.application.threadservice。

(2) 设计布局，在 res/layout 目录下的 main.xml 文件中，定义垂直线性布局 LinearLayout，在该布局中，设置一个 TextView 控件，两个 Button 控件。在该文件中编辑代码如下：

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <!-- 定义一个垂直线性布局 -->
3  <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
4      android:orientation="vertical"
5      android:layout_width="fill_parent"
6      android:layout_height="fill_parent"
7  >
8      <!-- 设置一个 TextView 控件 -->
9      <TextView  android:id="@+id/label"
10       android:layout_width="fill_parent"
11       android:layout_height="wrap_content"
12       android:text="@string/hello">
13     </TextView>
14     <!-- 设置一个 Button 控件，按钮名为"启动后台服务" -->
15     <Button android:id="@+id/start"
16         android:layout_width="wrap_content"
17         android:layout_height="wrap_content"
18         android:text="启动后台服务">
19     </Button>
20     <!-- 设置一个 Button 控件，按钮名为"停止后台服务" -->
21     <Button android:id="@+id/stop"
22         android:layout_width="wrap_content"
23         android:layout_height="wrap_content"
24         android:text="停止后台服务">
25     </Button>
26 </LinearLayout>
```

① 第 2 行至第 26 行定义一个垂直线性布局 LinearLayout。

② 第 9 行至第 13 行设置一个 TextView 控件。

③ 第 15 行至第 19 行、第 21 行至第 25 行分别设置两个按钮控件，按钮名分别为“启动后台服务”和“停止后台服务”。

(3) 在 com.application.threadservice 包下的 MyService.java 文件，定义 MyService 类继承自 Service 类，重写 onCreate()方法，使用 Toast 显示消息，创建 Thread 对象；重写 onStart()方法，使用 Toast 显示消息，启动线程；重写 onDestroy()方法，使用 Toast 显示消息，通知线程准备终止；实现 Runnable 接口，其中重写 run()方法。在该文件中编辑代码如下：

```
package com.application.threadservice;

import android.app.Service;
import android.content.Intent;
import android.os.IBinder;
import android.widget.Toast;

//定义 MyService 类继承自 Service 类
public class MyService extends Service{
    //声明 Thread 对象 testThread
    private Thread testThread;

    //重写 onCreate()方法,在该方法中, Toast 显示消息"(1) 调用 onCreate()",
    //创建 Thread 对象 testThread
    @Override
    public void onCreate() {
        super.onCreate();
        Toast.makeText(this, "(1) 调用 onCreate()", Toast.LENGTH_LONG).show();
        testThread = new Thread(null,backgroudWork,"TestThread");
    }

    //重写 onStart()方法,在该方法中,使用 Toast 显示消息"(2) 调用 onStart()",启动线程
    @Override
    public void onStart(Intent intent, int startId) {
          super.onStart(intent, startId);
          Toast.makeText(this, "(2) 调用 onStart()",Toast.LENGTH_SHORT).show();
          if (!testThread.isAlive()){
              testThread.start();
          }
    }

   //重写 onDestroy()方法,在该方法中,使用 Toast 显示消息"(3) 调用 onDestroy()",
   //调用 interrupt()方法通知线程准备终止
   @Override
   public void onDestroy() {
         super.onDestroy();
         Toast.makeText(this, "(3) 调用 onDestroy()",Toast.LENGTH_SHORT).show();
         testThread.interrupt();
   }

```

```
41      @Override
42      public IBinder onBind(Intent intent) {
43          return null;
44      }
45
46      //实现 Runnable 接口,其中重写 run()方法
47      private Runnable backgroudWork = new Runnable(){
48
49          @Override
50          public void run() {
51              try {
52                  while(!Thread.interrupted()){
53                      double randomDouble = Math.random();
54                      ThreadServiceActivity.UpdateGUI(randomDouble);
55                      Thread.sleep(1000);
56                  }
57              } catch (InterruptedException e) {
58                  e.printStackTrace();
59              }
60          }
61      };
62 }
```

① 第 9 行至第 62 行定义一个继承 Service 类的子类 MyService。

② 第 11 行,声明 Thread 对象 testThread。

③ 第 15 行至第 20 行,重写 onCreate()方法,在该方法中,第 18 行使用 Toast 显示消息"(1)调用 onCreate()",第 19 行创建 Thread 对象 testThread,在 Thread 的构造函数中,第 1 个参数表示线程组,第 2 个参数是需要执行的 Runnable 对象,第 3 个参数是线程的名称。

④ 第 24 行至第 30 行重写 onStart()方法,在该方法中,第 26 行使用 Toast 显示消息"(2)调用 onStart()",第 27 行至第 29 行启动线程 testThread。

⑤ 第 34 行至第 39 行重写 onDestroy()方法,在该方法中,第 37 行使用 Toast 显示消息"(3)调用 onDestroy()",第 38 行调用 interrupt()方法通知线程准备终止。

⑥ 第 47 行至第 61 行实现 Runnable 接口,其中,第 49 行至第 60 行重写 run()方法。

(4) 在 com.application.threadservice 包下的 ThreadServiceActivity.java 文件是界面的 Activity 文件,定义 ThreadServiceActivity 类继承自 Activity 类,创建 Handler 类的实例 handler,定义 UpdateGUI()方法,创建 Runnable 类的对象、并在其中重写 run()方法,重写 onCreate()方法。在该文件中编辑代码如下:

```
package com.application.threadservice;

import com.application.threadservice.R;
import android.app.Activity;
import android.content.Intent;
```

```
import android.os.Bundle;
import android.os.Handler;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;

//定义 ThreadServiceActivity 类继承自 Activity 类
public class ThreadServiceActivity extends Activity {

    //创建 Handler 类的实例 handler
    private static Handler handler = new Handler();
    private static TextView labelView = null;
    private static double randomDouble;

    //定义 UpdateGUI()方法,该方法是公共的界面更新方法
    public static void UpdateGUI(double refreshDouble){
        randomDouble = refreshDouble;
        handler.post(RefreshLable);
    }

    //创建 Runnable 类的对象 RefreshLable,其中重写 run()方法
    private static Runnable RefreshLable = new Runnable(){

        @Override
        public void run() {
            labelView.setText(String.valueOf(randomDouble));
        }
    };

   //重写 onCreate()方法,创建 Button 对象 startButton 和 stopButton,并分别绑定监听器
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        labelView = (TextView)findViewById(R.id.label);
        Button startButton = (Button)findViewById(R.id.start);
        Button stopButton = (Button)findViewById(R.id.stop);

        final Intent serviceIntent = new Intent(this, MyService.class);

       startButton.setOnClickListener(new Button.OnClickListener(){
            public void onClick(View view){
                startService(serviceIntent);
            }
```

```
51          });
52
53          stopButton.setOnClickListener(new Button.OnClickListener(){
54              public void onClick(View view){
55                  stopService(serviceIntent);
56              }
57          });
58      }
59 }
```

① 第 13 行至第 59 行定义一个继承 Activity 类的子类 ThreadServiceActivity。

② 第 16 行，创建 Handler 类的实例 handler，这个实例是私有的，外部代码不能直接调用这个实例。

③ 第 21 行至第 24 行，UpdateGUI()方法是公共的界面更新方法，将后台产生的数据 refreshDouble 传递到 UpdateGUI()方法，再调用 post()方法。

④ 第 27 行至第 33 行创建 Runnable 类的对象 RefreshLable，第 29 行至第 32 行在对象 RefreshLable 中重写 run()方法。

⑤ 第 36 行至第 58 行重写 onCreate()方法，创建 Button 对象 startButton 和 stopButton，并分别绑定监听器。

【运行结果】

在 Eclipse 中启动模拟器，然后运行项目 ThreadService，初始界面如图 6.10 所示。单击“启动后台服务”按钮后显示(1)和一个随机数如图 6.11 所示。



图 6.10　线程使用初始界面



图 6.11　单击“启动后台服务”按钮后显示“(1)调用 onCreate()”和一个随机数

然后显示(2)和另一个随机数如图 6.12 所示。单击“停止后台服务”按钮后显示效果如图 6.13 所示。



图 6.12　然后显示“(2)调用 onStart()”和另一个随机数



图 6.13　单击“停止后台服务”按钮后显示“(3)调用 onDestroy()”和另一个随机数

6.3　BroadcastReceiver 组件

BroadcastReceiver 是对广播消息进行过滤并响应的组件,不包含任何用户界面,其监听的事件源是其他组件。

BroadcastReceiver 是一种全局监听器,用于处理系统全局的广播信息,其处理机制是系统级别的,例如系统启动、闹钟、来电、系统时间改变、电池电量变化等,它与事件处理机制类似,但事件处理监听器 OnXxxListener()只是程序级别监听器,运行在指定程序的进程中,例如单击某个按钮的事件等。

BroadcastReceiver 类位于 android.content 包下。

BroadcastReceiver 运行机制如下:

(1) 发送广播:构建 Intent 对象,调用 sendBroadcast()方法。

(2) 接收广播服务:当其他组件通过 sendBroadcast()方法发送广播消息时,如果与注册时的 IntentFilte 过滤条件相匹配,就会调用 BroadcastReceiver 的 onReceive()方法,在该方法中响应事件。

1. 发送广播的方式

发送广播有以下三种方式：

1）使用 sendBroadcast 和 sendStickyBroadcast 发送广播

所有满足条件的 BroadcastReceiver 都会执行其 onReceive()方法来处理响应，是对广播消息进行过滤并响应的控件。

当有多个满足条件的 BroadcastReceiver 时，其 onReceive()方法的执行顺序是不定的。

2）使用 sendOrderBroadcast 发送广播

通过 sendOrderBroadcast 发送的 Intent，会根据 BroadcastReceiver 注册时 IntentFilter 设置的优先级顺序来执行 onReceive()方法。

对于相同优先级的 BroadcastReceiver，其 onReceive()方法的执行顺序是不定的。

3）使用 sendStickyBroadcast 发送广播

sendStickyBroadcast 发送广播与其他发送方式的不同之处是，Intent 在发送之后一直存在，并且在以后调用 registerReceive 注册相匹配的 Receive 时会把这个 Intent 直接返回给新注册的 Receive。

2. 接收广播服务的过程

（1）开发 BroadcastReveiver 类的子类，在其中重写 onReceive()方法。

（2）注册 BroadcastReceiver 对象。

BroadcastReceiver 注册方式有两种：动态注册和静态注册。

① 动态注册：通过调用 registerReceiver()方法来注册，代码如下：

```
MyReceiver receiver = new MyReceiver();          //创建相关对象
IntentFilter filter = new IntentFilter();
filter.addAction(DATE_CHANGED);
registerReceiver(receiver, filter);              //动态注册 BroadcastReceiver
```

② 静态注册：在 AndroidManifest.xml 中添加声明，代码如下：

```
<receiver android:name = ".MyReceiver">
<intent - filter >
    <action android:name = "android.intent.action.DATE_CHANGED"/>
    <category android:name = "android.intent.category.HOME"/>
    </intent - filter >
</receiver >
```

动态注册方式在代码中进行注册后，当应用程序关闭后，就不再进行监听。静态注册方式的特点：在应用程序安装之后，无论该应用程序是否处于活动状态，BroadcastReceiver 始终处于被监听状态。

在 AndroidManifest.xml 中为应用程序添加适当的权限，代码如下：

```
<uses - permission android:name = "android.permission.WRITE_SETTINGS"/>
```

我们注册的 BroadcastReceiver 并非一直在后台运行，一旦当事件或相关的 Intent 传

来，就会被系统调用，处理 onReceive()方法里的响应事件。

注意：在 onReceive()中执行的代码耗时不要超过 5s。

【例 6.4】 在用户界面发送广播消息后，内部的 BroadcastReceiver 将接收这个广播消息，并显示在用户界面的下方。

【解题思路】

创建一个 Intent，调用 sendBroadcast()方法将 Intent 携带的信息广播出去。在 AndroidManifest.xml 文件中注册一个 BroadcastReceiver，并使用 Intent 过滤器指定要处理的广播消息。当 Android 系统接收到与注册 BroadcastReceiver 匹配的广播消息时，会自动调用 BroadcastReceiver 接收广播消息，并调用 onReceive()方法进行处理。

【开发步骤和程序分析】

(1) 在 Eclipse 中创建一个 BroadcastMessage 应用项目，包名为 com.application.broadcastmessage。

(2) 设计布局，在 res/layout 目录下的 main.xml 文件中，定义垂直线性布局 LinearLayout，在该布局中，设置一个 TextView 控件，一个 EditText 控件，一个 Button 控件。在该文件中编辑代码如下：

```
<?xml version = "1.0" encoding = "utf - 8"?>
<! -- 定义一个垂直线性布局 -->
<LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = "vertical"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    >
   <! -- 设置一个 TextView 控件 -->
   <TextView android:id = "@ + id/label"
       android:layout_width = "fill_parent"
       android:layout_height = "wrap_content"
       android:text = "@string/hello"
    />
   <! -- 设置一个 EditText 控件 -->
   <EditText android:id = "@ + id/entry"
       android:text = ""
       android:layout_width = "fill_parent"
       android:layout_height = "wrap_content">
   </EditText>
   <! -- 设置一个 Button 控件，按钮名为"发送广播消息" -->
   <Button android:id = "@ + id/btn"
       android:layout_width = "wrap_content"
       android:layout_heig ht = "wrap_content"
       android:text = "发送广播消息"   >
   </Button>
</LinearLayout>
```

(3) 在 AndroidManifest.xml 文件中，编辑代码如下：

```
1  <?xml version = "1.0" encoding = "utf - 8"?>
2  < manifest xmlns:android = "http://schemas.android.com/apk/res/android"
3      package = "com.application.broadcastmessage"
4      android:versionCode = "1"
5      android:versionName = "1.0" >
6      < uses - sdk android:minSdkVersion = "14" />
7      < application android:icon = "@drawable/ic_launcher"
8        android:label = "@string/app_name" >
9         < activity android:label = "@string/app_name"
10              android:name = "com.application.broadcastmessage.BroadcastMessageActivity" >
11              < intent - filter >
12                  < action android:name = "android.intent.action.MAIN" />
13                  < category android:name = "android.intent.category.LAUNCHER" />
14              </intent - filter >
15          </activity >
16          < receiver android:name = "com.application.broadcastmessage.BroadcastService">
17              < intent - filter >
18                  < action android:name = "com.application.broadcastmessage" />
19              </intent - filter >
20          </receiver >
21      </application >
22  </manifest >
```

① 第 16 行建了一个< receiver >节点。

② 第 18 行声明了 Intent 过滤器的动作为“com.application.broadcastmessage”，与 BroadcastMessageActivity.java 文件中 Intent 的动作相一致，表明该 BroadcastReceiver 可以接收动作为“com.application.broadcastmessage”的广播消息。

(4) 在 com.application.broadcastmessage 包下的 BroadcastMessageActivity.java 文件中，定义 BroadcastMessageActivity 类继承自 Activity 类，构建 Intent 对象并调用 sendBroadcast()方法发送广播。在该文件中编辑代码如下：

```
package com.application.broadcastmessage;

import com.application.broadcastmessage.R;
import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;

```

```
12 //定义 BroadcastMessageActivity 类继承自 Activity 类
13 public class BroadcastMessageActivity extends Activity {
14     private EditText entryText ;
15     private Button button;
16
17     //重写 onCreate()方法
18     @Override
19     public void onCreate(Bundle savedInstanceState) {
20         super.onCreate(savedInstanceState);
21         setContentView(R.layout.main);
22         entryText = (EditText)findViewById(R.id.entry);
23         button = (Button)findViewById(R.id.btn);
24
25        //为 button 对象绑定事件监听器
26         button.setOnClickListener(new OnClickListener(){
27         public void onClick(View view){
28             //构建 Intent 对象并调用 sendBroadcast()方法发送广播
29             Intent intent = new Intent("com.application.broadcastmessage");
30             intent.putExtra("message", entryText.getText().toString());
31             sendBroadcast(intent);
32             }
33        });
34     }
35 }
```

① 第 13 行至第 35 行定义一个继承 Activity 类的子类 BroadcastMessageActivity。

② 第 29 行至第 31 行构建 Intent 对象并调用 sendBroadcast()方法发送广播。第 29 行构建 Intent 对象时将 com.application.broadcastmessage 作为识别广播的字符串标识，第 30 行添加额外信息，第 31 行调用 sendBroadcast()方法发送广播。

（5）在 com.application.broadcastmessage 包下的 BroadcastService.java 文件中，定义 BroadcastService 类继承自 BroadcastReceiver 类，重写 onReceive()方法，调用 getStringExtra()方法，获取标识为 message 的字符串数据，使用 Toast()方法将信息显示在界面。在该文件中编辑代码如下：

```
package com.application.broadcastmessage;

import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.widget.Toast;

//定义 BroadcastService 类继承自 BroadcastReceiver 类
```

```
9  public class BroadcastService extends BroadcastReceiver {
10
11     //重写 onReceive ()方法,调用 getStringExtra()方法,获取标识为 message 的字符串数据,
12     //使用 Toast()方法将信息显示在界面上
13     @Override
14     public void onReceive(Context context, Intent intent) {
15         String msg = intent.getStringExtra("message");
16         Toast.makeText(context, msg, Toast.LENGTH_SHORT).show();
17     }
18 }
```

① 第 9 行至第 18 行定义一个继承 BroadcastReceiver 类的子类 BroadcastService。

② 第 13 行至第 17 行重写 onReceive ()方法,当接收到 AndroidManifest.xml 文件定义的广播消息后,程序将自动调用 onReveive()方法进行消息处理。第 15 行调用 getStringExtra()方法,从 Intent 中获取标识为 message 的字符串数据,第 16 行使用 Toast()方法将信息显示在界面上。

【运行结果】

在 Eclipse 中启动模拟器,然后运行项目 BroadcastMessage,初始界面如图 6.14 所示,在编辑框中输入“Good!”的界面如图 6.15 所示。



图 6.14　发送广播消息初始界面



图 6.15　在编辑框中输入“Good!”

单击“发送广播消息”按钮后的显示效果如图 6.16 所示。



图 6.16　单击“发送广播消息”按钮后的界面

6.4　Notification

Android 系统提供的消息提示机制有 Toast 和 Notification。Toast 是一种快速的即时消息,消息内容简短,悬浮于应用程序的最上方。Notification 消息内容显示于手机的状态条中。有关 Toast 的内容,前面已做介绍,下面介绍有关 Notification 的内容。

Notification 类在 android.app 包下。Notification 消息内容显示在手机状态条中,无需 Activity。手机状态条位于手机屏幕最上方,用手指按住状态条往下拉,可打开状态条查看系统提示信息。

1. Notification 和 NotificationManager

Notification 有以下功能:除提示消息外,可创建新的状态栏图标,在扩展的状态条窗口显示额外的信息和其他提示形式信息,如闪烁 LED,让手机震动,发出声音(铃声、媒体库歌曲)等,也可以启动另一个 Intent。

所有的 Notification 都由 NotificationManager 来管理,通过 NotificationManager 显示出来。NotificationManager 常用方法如表 6.1 所示。

2. 使用 Notification 和 NotificationManager 的基本步骤

(1) 获取 NotificationManager 对象。

(2) 创建一个 Notification 对象。

创建 Notification 对象有以下两种格式:

表 6.1 NotificationManager 常用方法

方　　法	说　　明
cancel(int id)	取消以前显示的一个 Notification
cancelAll()	取消以前显示的所有 Notification
getSystemService(NOTIFICATION_SERVICE)	初始化一个 NotificationManager 对象
notify(int id, Notification notification)	把 Notification 持久地发送到状态条上

格式一：

```
Notification mynotification = new Notification(icon,ticker,when);
```

其中，icon 是显示在状态栏中的图标，一般通过资源 id 表示；ticker 是消息的文本内容；

when 是系统时间，一般可用 System.currentTimeMillis()获得。

格式二：

```
Notification mynotification = new Notification();
mynotification.icon = R.drawable.header;
mynotification.tickerText = getResources().getString(R.string.notification);
…
```

(3) 设置 Notification 的各个属性。

设置在状态条(Status Bar)显示的通知文本提示，设置发出提示音，设置手机震动，设置 LED 灯闪烁，设置对通知的单击事件处理。

(4) 发送通知。

下面介绍一个使用 BroadcastReceiver、Notification 和 NotificationManager 的例题。

【例 6.5】 单击一个按钮来发出一个广播通知，并将它显示在状态栏中，再单击另一个按钮，清除状态栏中的广播通知。

【解题思路】

定义两个类：BroadcastReceiverActivity 类和 MyBroadcast 类。

BroadcastReceiverActivity 类是一个 Activity 类，在其中构建需要广播的 Intent，然后在按钮的 onClick()方法中使用 sendBroadcast()方法发送广播，并且在该类的 onResume()方法中注册广播接收器。

MyBroadcast 类是一个继承 BroadcastReceiver 的子类，在该子类的 onReceive()方法中将传入的广播 Intent 中的信息发送给一个通知对象。

【开发步骤和程序分析】

(1) 在 Eclipse 中创建一个 BroadcastReceiverExample 应用项目，包名为 com.application.broadcastreceiverexample。

(2) 准备图片，将准备好的通知图标的图片资源复制到 res/drawable-mdpi 目录中。

(3) 准备字符串资源，在 res/values 目录下的 strings.xml 文件中编辑代码如下：

```
<?xml version = "1.0" encoding = "utf - 8"?>
```

```
<resources>
    <string name="hello">BroadcastReceiver Example !</string>
    <string name="app_name">BroadcastReceiverExample</string>
    <string name="btnsend">发送广播通知</string>
    <string name="btnclr">清除广播通知</string>
    <string name="brdcast">发出一个广播通知 !</string>
</resources>
```

（4）设计布局，在 res/layout 目录下的 main.xml 文件中，定义垂直线性布局 LinearLayout，在该布局中，设置一个 TextView 控件，两个 Button 控件。在该文件中编辑代码如下：

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <!-- 定义一个垂直线性布局 -->
3  <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
4      android:orientation="vertical"
5      android:layout_width="fill_parent"
6      android:layout_height="fill_parent" >
7      <!-- 设置一个 TextView 控件 -->
8      <TextView
9          android:layout_width="fill_parent"
10         android:layout_height="wrap_content"
11         android:text="@string/app_name"
12         android:gravity="center_horizontal"
13         android:padding="10sp"
14         android:textSize="8pt"
15         android:textColor="#00FF00" />
16     <!-- 设置一个 Button 控件 -->
17     <Button android:id="@+id/BTN_SEND"
18         android:layout_width="wrap_content"
19         android:layout_height="wrap_content"
20         android:layout_gravity="center_horizontal"
21         android:text="@string/btnsend" />
22     <!-- 设置一个 Button 控件 -->
23     <Button android:id="@+id/BTN_CLEAR_NOTIFICATION"
24         android:layout_width="wrap_content"
25         android:layout_height="wrap_content"
26         android:layout_gravity="center_horizontal"
27        android:text="@string/btnclr" />
28 </LinearLayout>
```

① 第 2 行至第 28 行定义一个垂直线性布局 LinearLayout。

② 第 8 行至第 15 行设置一个 TextView 控件。

③ 第 17 行至第 21 行、第 23 行至第 27 行分别设置一个 Button 控件，两个按钮显示的文本来自 strings.xml 文件。

(5) 在 com.application.broadcastreceiverexample 包下的 BroadcastReceiverActivity.java 文件中，定义 BroadcastMessageActivity 类继承自 Activity 类并实现 OnClickListener 接口，定义“发送广播通知”按钮对象和“清除广播通知”按钮对象，重写 onClick()方法，当单击“发送广播通知”按钮时执行 doSend()方法，广播发送一个 Intent 对象 sendIntent。当单击“清除广播通知”按钮时执行 doclearnotification()方法，清除 ID 号为 NOTIFICATION_ID 的通知。在该文件中编辑代码如下：

```
package com.application.broadcastreceiverexample;

import com.application.broadcastreceiverexample.R;
import android.app.Activity;
import android.app.NotificationManager;
import android.content.Intent;
import android.content.IntentFilter;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;

//定义 BroadcastReceiverActivity 类继承自 Activity 类并实现 OnClickListener 接口
public class BroadcastReceiverActivity extends Activity implements OnClickListener {

    public static final String ACTION = "com.application.broadcastreceiverexample";
    public static final String INTENT_EXTRAS_TITLE = "BroadcastReciver Activity";
    private MyBroadcast mReceiver = null;
    private IntentFilter mIntentFilter = null;

    //重写 onCreate()方法
    @Override
    protected void onCreate(Bundle icicle) {
        super.onCreate(icicle);
        setContentView(R.layout.main);
        //分别定义"发送广播通知"按钮对象 btnSend 和"清除广播通知"按钮对象 btnClear
        Button btnSend = (Button)findViewById(R.id.BTN_SEND);
        Button btnClear = (Button)findViewById(R.id.BTN_CLEAR_NOTIFICATION);
        btnSend.setOnClickListener(this);
        btnClear.setOnClickListener(this);
    }

  //重写 onResume()方法，在这个方法中使用 registerReceiver()方法注册广播接收器
  @Override
  protected void onResume() {
        mReceiver = new MyBroadcast();
        mIntentFilter = new IntentFilter();
```

```
        mIntentFilter.addAction(ACTION);
        this.registerReceiver(mReceiver, mIntentFilter);
        super.onResume();
    }

//重写 onDestroy()方法,在这个方法中使用 unregisterReceiver()方法注销广播接收器
@Override
protected void onDestroy() {
        if(mReceiver != null) {
            this.unregisterReceiver(mReceiver);
        };
        super.onDestroy();
}

//重写 onClick()方法
  @Override
  public void onClick(View v) {
        switch(v.getId() ) {
            case R.id.BTN_SEND: {
                doSend();
                break;
            }
            case R.id.BTN_CLEAR_NOTIFICATION: {
                doclearnotification();
                break;
            }
        }
  }

//定义 doSend()方法,广播发送一个 Intent 对象 sendIntent
  private void doSend() {
        Intent sendIntent = new Intent(ACTION);
        sendIntent.putExtra("extit", INTENT_EXTRAS_TITLE);
        sendIntent.putExtra("exmsg",getString(R.string.brdcast).toString());
        this.sendBroadcast(sendIntent);
  }

//定义 doclearnotification()方法,清除 ID 号为 NOTIFICATION_ID 的通知
  private void doclearnotification() {
        NotificationManager notificationManager = (NotificationManager)
        this.getSystemService(android.content.Context.NOTIFICATION_SERVICE);
       notificationManager.cancel(MyBroadcast.NOTIFICATION_ID);
  }
}
```

① 第 14 行至第 81 行定义一个实现 OnClickListener 接口的实现类 BroadcastReceiverActivity，这个类继承 Activity 类。

② 第 27 行至第 28 行分别定义“发送广播通知”按钮 btnSend 和“清除广播通知”按钮 btnClear。

③ 第 34 行至第 41 行重写 onResume()方法，在这个方法中使用 registerReceiver()方法注册广播接收器。

④ 第 44 行至第 50 行重写 onDestroy()方法，在这个方法中使用 unregisterReceiver()方法注销广播接收器。

⑤ 第 53 行至第 65 行重写 onClick()方法。当单击“发送广播通知”按钮时执行 doSend()方法，第 68 行至第 73 行定义该方法，广播发送一个 Intent 对象 sendIntent。当单击“清除广播通知”按钮时执行 doclearnotification()方法，第 76 行至第 80 行定义该方法，清除 ID 号为 NOTIFICATION_ID 的通知。

(6) 在 com.application.broadcastreceiverexample 包下的 MyBroadcast.java 文件中，定义 MyBroadcast 类继承自 BroadcastReceiver 类，重写 onReceive ()方法，从传入的 Intent 对象中取出该 intent 附加的信息，获取 NotificationManager 的实例，实例化 Notification，获取 PendingIntent 对象，设置事件信息，发出 ID 号为 NOTIFICATION_ID 的通知。在该文件中编辑代码如下：

```
package com.application.broadcastreceiverexample;

import com.application.broadcastreceiverexample.R;
import android.app.Notification;
import android.app.NotificationManager;
import android.app.PendingIntent;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;

//定义 MyBroadcast 类继承自 BroadcastReceiver 类
public class MyBroadcast extends BroadcastReceiver {
    Context context;
    public static int NOTIFICATION_ID = 21321;

    //重写 onReceive ()方法
    @Override
    public void onReceive(Context context, Intent intent) {
        this.context = context;
    //从传入的 Intent 对象中取出该 intent 附加的信息
        String titstr = intent.getStringExtra("extit");
        String msgstr = intent.getStringExtra("exmsg");
        //获取 NotificationManager 的实例
        NotificationManager notificationmanager = (NotificationManager) context
```

```
25                 .getSystemService(android.content.Context.NOTIFICATION_SERVICE);
26             //实例化 Notification
27             Notification notification = new Notification(R.drawable.icon,
28                 titstr, System.currentTimeMillis());
29             //获取 PendingIntent 对象
30             PendingIntent pintent = PendingIntent.getActivity(context, 0,
31                 new Intent(context, BroadcastReceiverActivity.class), 0);
32             //设置事件信息
33             notification.setLatestEventInfo(context, msgstr, null,pintent);
34             //发出 ID 号为 NOTIFICATION_ID 的通知
35             notificationmanager.notify(NOTIFICATION_ID, notification);
36         }
37 }
```

① 第 11 行至第 29 行定义一个继承 BroadcastReceiver 类的子类 MyBroadcast。

② 第 16 行至第 28 行重写 onReceive ()方法。第 18 行至第 19 行从传入的 Intent 对象中取出该 intent 附加的信息,第 20 行至第 21 行获取 NotificationManager 的实例,第 22 行至第 23 行实例化 Notification,第 24 行至第 25 行获取 PendingIntent 对象,第 26 行设置事件信息,第 27 行发出 ID 号为 NOTIFICATION_ID 的通知。

【运行结果】

在 Eclipse 中启动模拟器,然后运行项目 BroadcastReceiverExample,初始界面如图 6.17 所示。单击“发送广播通知”按钮后,出现状态信息,如图 6.18 所示。





图 6.17 BroadcastReceiver 应用举例初始界面



图 6.18 单击“发送广播通知”按钮后的界面

拉下状态条后显示通知信息，如图 6.19 所示。单击“清除广播通知”按钮后，状态条中的状态信息被清除，如图 6.20 所示。



图 6.19　拉下状态条后显示的通知信息



图 6.20　单击“清除广播通知”按钮后的界面

6.5　花卉图片的幻灯片展示

为了更好地理解和综合应用后台服务组件，下面介绍一个使用 Handler 机制以幻灯片的形式显示花卉图片，每张图片停留 4 秒钟。

【例 6.6】　花卉图片的幻灯片展示

【解题思路】

定义两个类，一个类继承 Activity 类，在这个类中创建 Handler 对象，重写 handleMessage()方法，在该方法中根据消息的值确定 UI 的显示内容。另一个类继承 Thread 类，重写 run()方法，在该方法中使用 sendEmptyMessage()方法向 Handler 发送消息。

【开发步骤和程序分析】

(1) 在 Eclipse 中创建一个 SlideExample 应用项目，包名为 com.application.slideexample。

(2) 准备图片，将准备好的通知图标的图片资源复制到 res/drawable-mdpi 目录中。

(3) 准备字符串资源，在 res/values 目录下的 strings.xml 文件中编辑代码如下：

```
<?xml version = "1.0" encoding = "utf - 8"?>
< resources >
```

```
3        <string name = "hello"> Slide Example !</string>
4        <string name = "app_name"> SlideExample </string>
5        <string name = "title">花卉展示</string>
6        <string name = "photo01">梨花</string>
7        <string name = "photo02">柳枝</string>
8        <string name = "photo03">樱花</string>
9        <string name = "photo04">海棠花 </string>
10 </resources>
```

① 第 2 行至第 11 行定义一个垂直线性布局 LinearLayout。

② 第 5 行至第 10 行设置一个 TextView 控件。

(4) 设计布局,在 res/layout 目录下的 main. xml 文件中,定义相对布局 RelativeLayout,在该布局中,设置一个 TextView 控件,一个 ImageView 控件,一个 Button 控件。在该文件中编辑代码如下:

```
<?xml version = "1.0" encoding = "utf - 8"?>
<! -- 定义一个相对布局 RelativeLayout -->
<RelativeLayout
        xmlns:android = "http://schemas.android.com/apk/res/android"
        android:lay   out_width = "fill_parent"
        android:layout_height = "fill_parent"
        android:layout_margin = "2dip">"
    <! -- 设置一个 TextView 控件,用于显示标题 -->
    <TextView
          android:id = "@ + id/picTitle"
         android:layout_width = "wrap_content"
         android:layout_height = "50dip"
         android:layout_alignParentTop = "true"
         android:layout_centerHorizontal = "true"
         android:gravity = "center_vertical"
         android:textSize = "22sp"
         android:textColor = "#ffffff00"
         android:text = "@string/title" />
    <! -- 设置一个 ImageView 控件,用于显示花卉图片 -->
    <ImageView
         android:id = "@ + id/myPic"
         android:layout_width = "fill_parent"
         android:layout_height = "fill_parent"
         android:layout_below = "@id/picTitle"
        android:layout_alignParentTop = "true"
        android:layout_alignParentLeft = "true"
        android:src = "@drawable/photo01" />
    <! -- 设置一个 TextView 控件,用于显示花卉图片的说明 -->
    <TextView
         android:id = "@ + id/picName"
```

```
31          android:layout_width = "fill_parent"
32          android:layout_height = "50dip"
33          android:layout_alignParentBottom = "true"
34          android:layout_alignParentLeft = "true"
35          android:gravity = "center"
36          android:background = "#55ffff00"
37          android:textSize = "18sp"
38          android:text = "@string/photo01" />
39  </RelativeLayout>
```

① 第 3 行至第 39 行定义一个相对布局 RelativeLayout。

② 第 9 行至第 18 行设置一个 TextView 控件，用于显示标题“花卉展示”。

③ 第 20 行至第 27 行设置一个 ImageView 控件，用于显示花卉图片。

④ 第 29 行至第 38 行设置一个 TextView 控件，用于显示花卉图片的说明。

（5）在 com.application.slideexample 包下的 SlideExampleActivity.java 文件中，定义 SlideExampleActivity 类继承 Activity 类，创建一个 Handler 对象，重写 handleMessage()方法，根据传入形参的 what 值，确定 ImageView 和 TextView 的显示内容，重写 onCreate 方法，初始化 MyThread 线程，启动线程。在该文件中编辑代码如下：

```
package com.application.slideexample;

import com.application.slideexample.R;
import android.app.Activity;
import android.os.Bundle;
import android.os.Handler;
import android.os.Message;
import android.widget.ImageView;
import android.widget.TextView;

//定义一个类 SlideExampleActivity 继承 Activity 类
public class SlideExampleActivity extends Activity {
    ImageView myPicture;
    TextView myPicname;
    //创建一个 Handler 对象
    Handler myHandler  =  new Handler(){

       //重写 handleMessage()方法，在该方法中，根据传入形参的 what 值，
       //确定 ImageView 和 TextView 的显示内容
        @Override
        public void handleMessage(Message msg) {
            switch(msg.what){
            case 0:
                myPicture.setImageResource(R.drawable.photo01);
                myPicname.setText(R.string.photo01);
                break;
```

```
27                  case 1:
28                      myPicture.setImageResource(R.drawable.photo02);
29                      myPicname.setText(R.string.photo02);
30                      break;
31                  case 2:
32                      myPicture.setImageResource(R.drawable.photo03);
33                      myPicname.setText(R.string.photo03);
34                      break;
35                  case 3:
36                      myPicture.setImageResource(R.drawable.photo04);
37                      myPicname.setText(R.string.photo04);
38                      break;
39                  }
40                  super.handleMessage(msg);
41              }
42          };
43
44      //重写 onCreate 方法
45      @Override
46          public void onCreate(Bundle savedInstanceState) {/
47              super.onCreate(savedInstanceState);
48              setContentView(R.layout.main);
49              myPicture = (ImageView) findViewById(R.id.myPic);
50              myPicname = (TextView) findViewById(R.id.picName);
51              //初始化 MyThread 线程
52              SlideThread myThread = new SlideThread(this);
53              //启动线程
54              myThread.start();
55      }
56 }
```

① 第 12 行至第 56 行定义一个类 SlideExampleActivity 继承 Activity 类，第 16 行创建一个 Handler 对象。

② 第 20 行至第 42 行重写 handleMessage()方法，在该方法中，根据传入形参的 what 值，确定 ImageView 和 TextView 的显示内容。

③ 第 45 行至第 55 行重写 onCreate 方法。第 48 行设置当前的用户界面，第 49 行获取 ImageView 的引用，第 50 行获取 TextView 的引用，第 52 行初始化 MyThread 线程，第 54 行启动线程。

(6) 在 com.application.slideexample 包下的 SlideThread.java 文件中，定义 SlideThread 类继承 Thread 类，重写 run()方法，使用 sendEmptyMessage()方法发送消息，每隔 4 秒向 Activity 的 myHandler 发送一个消息值。在该文件中编辑代码如下：

```
1  package com.application.slideexample;
```

```
2
3 //定义 SlideThread 类继承 Thread 类
4  public class SlideThread extends Thread{
5       SlideExampleActivity activity;
6       int what = 1;
7
8       //在 SlideThread 构造器中,获取 activity 的引用
9       public SlideThread(SlideExampleActivity activity){
10           this.activity = activity;
11       }
12
13     //重写 run()方法,使用 sendEmptyMessage()方法发送消息,
14     //每隔 4 秒向 Activity 的 myHandler 发送一个消息值
15      @Override
16       public void run() {
17          while(true){
18              activity.myHandler.sendEmptyMessage((what++) % 4);
19              try{
20                  Thread.sleep(4000);
21              }
22             catch(Exception e){
23                 e.printStackTrace();
24             }
25          }
26      }
27 }
```

① 第 4 行至第 27 行定义一个继承 Thread 类的子类 SlideThread。第 6 行设置发送消息的 what 值,在第 9 行至第 11 行的构造器中,获取 activity 的引用。

② 第 15 行至第 26 行重写 run()方法。第 17 行至第 25 行的循环中,使用 sendEmptyMessage()方法发送消息,每隔 4 秒向 Activity 的 myHandler 发送一个消息值,这个值由表达式(what++)%4 计算,其计算结果为 0,1,2,3。

【运行结果】

在 Eclipse 中启动模拟器,然后运行项目 SlideExample,每隔 4 秒逐一显示花卉展示中的图片,循环展示,如图 6.21 和图 6.22 所示。



图 6.21　显示花卉图片 1



图 6.22　显示花卉图片 2

6.6　小　　结

本章主要介绍了以下内容：

(1) Service 组件适用于没有用户界面，并长时间在后台运行的应用，例如播放音乐、检测 SD 卡上文件的变化、后台数据计算和发送通知等。

Service 的生命周期方法有 onCreate()、onStart()和 onDestroy()等。

Service 不能自己启动，需要通过某一个 Activity 启动，也可由其他 Service 或者 BroadcastReceiver 启动。Service 的启动方式有两种：通过 startService 启动和通过 bindService 启动。

(2) 在启动模式下，有两种方法启动 Service：显式启动和隐式启动。

绑定模式下的 Service 使用 bindService()方法启动 Service 和使用 unbindService()方法取消绑定。

(3) 创建线程步骤如下：

① 实现 Runnable 接口，并重写 run()方法，在 run()方法中放置代码的主体部分。

② 创建 Thread 对象，并将 Runnable 对象作为参数传递给 Thread 对象。

③ 调用 start()方法启动线程。

当线程在 run()方法返回后，线程就自动终止了，也可以调用 stop()在外部终止线程，最好的方法是通知线程自行终止，一般调用 interrupt()方法通知线程准备终止。

(4) BroadcastReceiver 是对广播消息进行过滤并响应的组件，不包含任何用户界面，其监听的事件源是其他组件。

BroadcastReceiver 运行机制如下：

① 发送广播：构建 Intent 对象，调用 sendBroadcast()方法。

② 接收广播服务：当其他组件通过 sendBroadcast()方法发送广播消息时，如果与注册时的 IntentFilte 过滤条件相匹配，就会调用 BroadcastReceiver 的 onReceive()方法，在该方法中响应事件。

(5) Android 系统提供的消息提示机制有 Toast 和 Notification。Toast 是一种快速的即时消息，消息内容简短，悬浮于应用程序的最上方。Notification 消息内容显示于手机的状态条中。

使用 Notification 和 NotificationManager 的基本步骤如下：

① 获取 NotificationManager 对象。

② 创建一个 Notification 对象。

③ 设置 Notification 的各个属性。

④ 发送通知。

习　题　6

一、选择题

6.1　Service 的生命周期方法不包括________。

A. onDestroy()　　B. onCreate()　　C. onResume()　　D. onStart()

6.2　如果 Service 已经启动，再次启动 Service 时，将直接执行________方法。

A. onDestroy()　　B. onCreate()　　C. onResume()　　D. onStart()

6.3　当服务不再使用且即将被销毁时，系统会调用________方法。

A. onStart()　　B. onDestroy()　　C. onBind ()　　D. onCreate()

6.4　下面选项正确的是________。

A. 一个服务只会创建一次，销毁一次　　B. 一个服务可以创建多次，销毁一次

C. 一个服务可以创建一次，销毁多次　　D. 一个服务可以创建多次，销毁多次

6.5　在一个 Service 的方法中，下面选项正确的是________。

A. onCreate()和 onDestroy()方法和 onStart()都只能调用一次

B. onCreate()和 onDestroy()方法可以调用多次，onStart()方法只能调用一次

C. onCreate()和 onDestroy()方法只能调用一次，onStart()方法可以调用多次

D. onCreate()、onDestroy()和 onStart()方法都可以调用多次

6.6　BroadcastReceiver 发送广播的方式不包括________。

A. sendBroadcast　　B. sendIntent

C. sendOrderBroadcast　　D. sendStickyBroadcast

二、填空题

6.7　Service 的生命周期方法有 onStart()、onDestroy()和________等。

6.8　Service 可通过某一个 Activity 启动，也可由其他 Service 或者________启动。

6.9　Service 的启动方式有两种：通过 startService 启动和通过________启动。

6.10　在启动模式下，有两种方法启动 Service：显式启动和________。

6.11　绑定模式下的 Service 使用 bindService()方法启动 Service 和使用________方法取消绑定。

6.12　BroadcastReceiver 是对广播消息进行过滤并响应的组件，不包含任何用户界面，其监听的事件源是________。

6.13　Android 系统提供的消息提示机制有 Toast 和________。

三、问答题

6.14　什么是 Service？ Service 与 Activity 有何区别？

6.15　简述 Service 生命周期回调方法和启动方式。

6.16　什么是 BroadcastReceive？ 简述其运行机制。

6.17　BroadcastReceiver 注册方式有哪两种？

6.18　发送广播有方式有哪些？ 各有何特点？

6.19　Android 系统提供的消息提示机制有哪些？ 各有何特点？

四、应用题

6.20　使用 Service 实现比较两个整数大小的功能，在输入两个整数后，输出较大的整数。

6.21　定义两个按钮，单击一个按钮发出一个广播通知，单击另一个按钮清除广播通知。

第 7 章　数据存储

本章要点

- SharedPreferences(偏好设置)是一种存储简单信息的机制,通过调用其方法可以实现键-值对的保存和读取,并能够实现不同应用程序间的数据共享。
- Android 支持标准的 Java I/O 读写方法,可以建立和访问程序自身建立的数据文件,可以将文件保存在 SD 卡等外部存储设备中,也可以访问保存在资源目录中的原始文件和 XML 文件。
- SQLite 是 Android 自带的轻量级关系数据库,使用资源少,运行高效可靠,可移植性强。
- SQLiteDatabase 类的方法 insert()、update()、delete()和 query(),可执行 SQL 语句对数据库进行插入、更新、删除和查询等操作。
- SQLiteOpenHelper 类是一个重要的帮助类,这个帮助类可以辅助创建、更新和打开数据库。
- ContentProvider(数据提供器)支持在多个应用程序中存储和读取数据,是应用程序之间共享数据的一种接口机制,提供一套标准接口用来获取和操作数据。

程序是数据的输入、处理和输出的过程,数据存储是程序的最基本的问题,Android 为应用开发人员提供了多种数据存储方式,本章介绍 SharedPreferences、文件存储、SQLite 数据库、数据共享等内容。

7.1　SharedPreferences

SharedPreferences 提供了一种轻量级的数据存取方法,它是 Android 的一种存储简单信息的机制。

通过 SharedPreferences 开发人员可以键-值对(Name-Value Pair)的方式存储,而且 SharedPreferences 完全屏蔽了对文件系统的操作过程,仅通过调用 SharedPreferences 中的方法就可以实现键-值对的保存和读取,SharedPreferences 不仅能够保存数据,还能够实现不同应用程序间的数据共享。

1. SharedPreferences 对象

每个应用程序都有一个 SharedPreferences 对象,通过 getSharedPreferences(String name, int mode)方法获取 SharedPreferences 对象。

其中,第一个参数 name 为本组件的配置文件名称。第二个参数 mode 为设置

SharedPreferences 对象的访问模式。

SharedPreferences 支持三种访问模式：

- MODE_PRIVATE：私有访问模式，仅创建 SharedPreferences 的程序有权限对其进行读取或写入。
- MODE_WORLD_READABLE：全局读访问模式，不仅创建程序可以对其进行读取或写入，其他应用程序也具有读取操作的权限，但没有写入操作的权限。
- MODE_WORLD_WRITEABLE：全局写访问模式，所有程序都可以对其进行写入操作，但没有读取操作的权限。

2. SharedPreferences 对象读写

1）SharedPreferences 对象的数据读取

通过 SharedPreferences 对象的键 key，获取到对应 key 的键值。对于不同类型的键值有不同的方法：getString、getBoolean、getInt、getFloat、getLong。例如：

```
SharedPreferences sharedPreferences = getSharedPreferences(PREFERENCE_NAME, MODE);
String name = sharedPreferences.getString("Name","Mike");
int age = sharedPreferences.getInt("Age", 21);
float height = sharedPreferences.getFloat("Height",1.70f);
```

2）SharedPreferences 对象的数据存入

通过 SharedPreferences 对象的编辑器对象 Editor 来实现的。通过编辑器方法设置键值，然后调用 commit()提交设置，写入 XML 文件。例如：

```
SharedPreferences.Editor editor = sharedPreferences.edit();
editor.putString("Name", "Mike");
editor.putInt("Age", 21);
editor.putFloat("Height", 1.70f);
editor.commit();
```

【例 7.1】 通过 SharedPreferences 存储举例，介绍 SharedPreferences 的文件保存位置和格式。

【解题思路】

用户在界面上的输入信息，在 Activity 关闭时通过 SharedPreferences 进行保存。当应用程序重新开启时，再通过 SharedPreferences 将信息读取出来，并重新呈现在用户界面上。

【开发步骤和程序分析】

(1) 在 Eclipse 中创建一个 PreferenceExample 应用项目，包名为 com.application.preferenceexample。

(2) 设计布局，在 res/layout 目录下的 main.xml 文件中，定义相对布局 RelativeLayout，在该布局中，设置三个 EditText 控件，设置三个 TextView 控件。在该文件中编辑代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<!-- 定义一个相对布局 RelativeLayout -->
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
```

```
    android:id = "@ + id/RelativeLayout01"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content" >
        <! -- 设置一个 EditText 控件,用于设置姓名的文本输入 -->
        < EditText android:id = "@ + id/name"
            android:text = ""
            android:layout_width = "280dip"
            android:layout_height = "wrap_content"
            android:layout_alignParentRight = "true"
            android:layout_marginLeft = "10dip" >
        </EditText >
        <! -- 设置一个 TextView 控件,用于设置姓名标签 -->
        < TextView android:id = "@ + id/name_label"
            android:text = "姓名: "
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:layout_alignParentLeft = "true"
            android:layout_toRightOf = "@id/name"
            android:layout_alignBaseline = "@ + id/name">
        </TextView >
        <! -- 设置一个 EditText 控件,用于设置性别的文本输入 -->
        < EditText android:id = "@ + id/sex"
            android:text = ""
            android:layout_width = "280dip"
            android:layout_height = "wrap_content"
            android:layout_alignParentRight = "true"
            android:layout_marginLeft = "10dip"
            android:layout_below = "@id/name" >
        </EditText >
        <! -- 设置一个 TextView 控件,用于设置性别标签 -->
        < TextView android:id = "@ + id/sex_label"
            android:text = "性别: "
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:layout_alignParentLeft = "true"
            android:layout_toRightOf = "@id/sex"
            android:layout_alignBaseline = "@ + id/sex" >
        </TextView >
        <! -- 设置一个 EditText 控件,用于设置总学分的文本输入 -->
        < EditText android:id = "@ + id/totalcredits"
            android:layout_width = "280dip"
            android:layout_height = "wrap_content"
            android:layout_alignParentRight = "true"
            android:layout_marginLeft = "10dip"
            android:layout_below = "@id/sex"
```

```
49              android:numeric = "integer">
50          </EditText >
51          <! -- 设置一个 TextView 控件,用于设置总学分标签 -->
52          < TextView android:id = "@ + id/totalcredits_label"
53              android:text = "总学分: "
54              android:layout_width = "wrap_content"
55              android:layout_height = "wrap_content"
56              android:layout_alignParentLeft = "true"
57              android:layout_toRightOf = "@id/totalcredits"
58              android:layout_alignBaseline = "@ + id/totalcredits">
59          </TextView >
60      </RelativeLayout >
```

① 第 3 行至第 60 行定义一个相对布局 RelativeLayout。

② 第 8 行至第 14 行、第 16 行至第 23 行分别设置一个 EditText 控件、一个 TextView 控件,用于设置姓名的文本输入及标签。

③ 第 25 行至第 32 行、第 34 行至第 41 行分别设置一个 EditText 控件、一个 TextView 控件,用于设置性别的文本输入及标签。

④ 第 43 行至第 50 行、第 52 行至第 59 行分别设置一个 EditText 控件、一个 TextView 控件,用于设置总学分的文本输入及标签。

(3) 在 com. application. preferenceexample 包下的 PreferenceExampleActivity. java 文件中,定义 PreferenceExampleActivity 类继承 Activity 类,重写 onStart()方法,在这个方法中调用 loadSharedPreferences()方法,读取保存在 SharedPreferences 中的姓名、性别、总学分信息,并显示在用户界面上;重写 onStop ()方法,在这个方法中调用 saveSharedPreferences()方法,保存用户界面上的信息。在该文件中编辑代码如下:

```
package com.application.preferenceexample;

import com.application.preferenceexample.R;
import android.app.Activity;
import android.content.Context;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.widget.EditText;
//定义 PreferenceExampleActivity 类继承 Activity 类
public class PreferenceExampleActivity extends Activity {
    //定义 EditText 类的属性 nameText、sexText、totalcreditsText,
    //用于接收姓名、性别、总学分
    private EditText nameText;
    private EditText sexText;
    private EditText totalcreditsText;
    public static final String PREFERENCE_NAME = "SavePrefs";
    public static int MODE = Context.MODE_WORLD_READABLE
      + Context.MODE_WORLD_WRITEABLE;
```

```

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        //分别获取姓名 EditText、性别 EditText、总学分 EditText
        nameText = (EditText)findViewById(R.id.name);
        sexText = (EditText)findViewById(R.id.sex);
        totalcreditsText = (EditText)findViewById(R.id.totalcredits);
    }

    //重写 onStart()方法,在这个方法中调用 loadSharedPreferences()方法,读取保存在
    //SharedPreferences 中的姓名、性别、总学分信息,并显示在用户界面上
    @Override
    public void onStart(){
        super.onStart();
        loadSharedPreferences();
    }

    //重写 onStop ()方法,在这个方法中调用 saveSharedPreferences()方法,
    //保存用户界面上的信息
    @Override
    public void onStop(){
        super.onStop();
        saveSharedPreferences();
    }

    //定义 loadSharedPreferences()方法,从 SharedPreferences 中读取姓名、性别、总学分
    private void loadSharedPreferences(){
        SharedPreferences sharedPreferences =
            getSharedPreferences(PREFERENCE_NAME, MODE);
        String name = sharedPreferences.getString("Name","David");
        String sex = sharedPreferences.getString("Sex", "male");
        int totalcredits = sharedPreferences.getInt("Totalcredits",52);
        nameText.setText(name);
        sexText.setText(String.valueOf(sex));
        totalcreditsText.setText(String.valueOf(totalcredits));
    }

    //定义 saveSharedPreferences()方法,将姓名、性别、总学分存入 SharedPreferences
    private void saveSharedPreferences(){
        SharedPreferences sharedPreferences =
            getSharedPreferences(PREFERENCE_NAME, MODE);
        SharedPreferences.Editor editor = sharedPreferences.edit();
        editor.putString("Name", nameText.getText().toString());
```

```
64            editor.putInt("Sex", Integer.parseInt(sexText.getText().toString()));
65            editor.putFloat("Totalcredits",
66                 Float.parseFloat(totalcreditsText.getText().toString()));
67            editor.commit();
68        }
69 }
```

① 第 10 行至第 69 行定义一个类 PreferenceExampleActivity 继承 Activity 类。

② 第 13 行至第 15 行分别定义 EditText 类的属性 nameText、sexText、totalcreditsText，用于接收姓名、性别、总学分。

③ 第 25 行至第 27 行分别获取姓名 EditText、性别 EditText、总学分 EditText。

④ 第 32 行至第 36 行重写 onStart()方法，在这个方法中调用 loadSharedPreferences()方法，读取保存在 SharedPreferences 中的姓名、性别、总学分信息，并显示在用户界面上。

⑤ 第 40 行至第 44 行重写 onStop ()方法，在这个方法中调用 saveSharedPreferences()方法，保存用户界面上的信息。

⑥ 第 47 行至第 56 行定义 loadSharedPreferences()方法，从 SharedPreferences 中读取姓名、性别、总学分。

⑦ 第 58 行至第 68 行定义 saveSharedPreferences()方法，将姓名、性别、总学分存入 SharedPreferences。

【运行结果】

在 Eclipse 中启动模拟器，然后运行项目 PreferenceExample，SharedPreferences 存储举例界面如图 7.1 所示。



图 7.1　SharedPreferences 存储举例界面

SharedPreferences 产生的文件保存在/data/data/com. application. preferenceexample/shared_prefs 目录下，如图 7.2 所示。

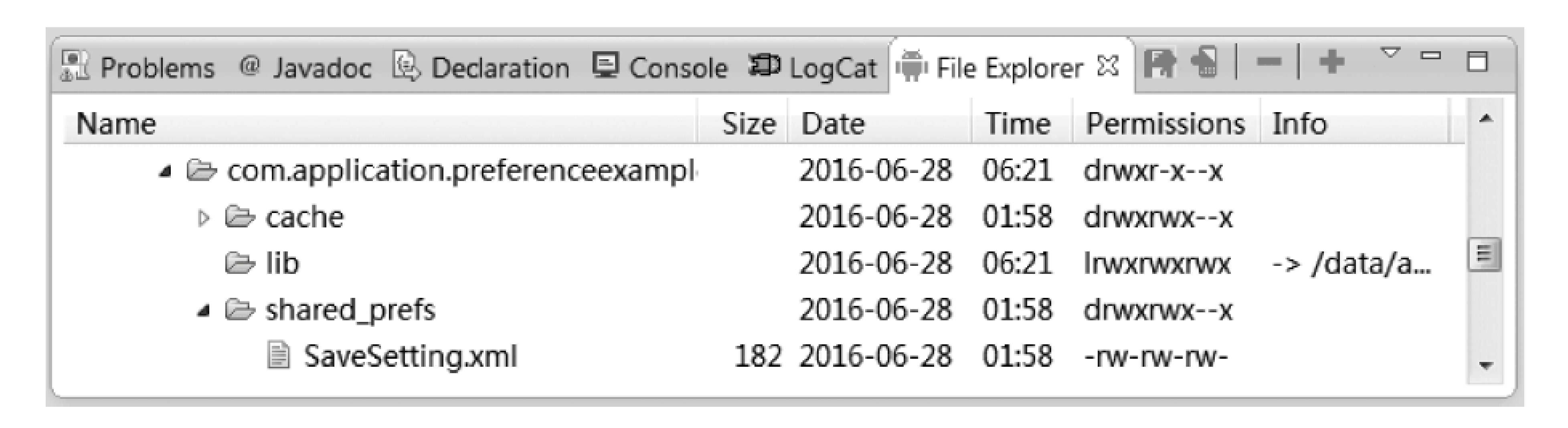


图 7.2　保存 SharedPreferences 的文件

7.2 文件存储

文件存储常用于存储一些声音、视频、图像等媒体文件。

Android 支持标准的 Java I/O 读写方法，可以建立和访问程序自身建立的私有文件，可以将文件保存在 SD 卡等外部存储设备中，也可以访问保存在资源目录中的原始文件和 XML 文件。

7.2.1 数据文件的存取操作

Android 系统允许应用程序创建的文件是用于自身访问的私有文件，文件保存在内部存储器上，位于 Android 系统的目录：data/data/< package name >/files。

下面介绍两个方法：openFileOutput()和 openFileInput()。

1. openFileOutput()方法

openFileOutput()方法打开一个指定的文件，用于写入数据。如果指定的文件存在，直接打开文件写入数据；如果指定的文件不存在，则创建一个新的文件。

openFileOutput()方法的格式如下：

```
public FileOutputStream openFileOutput(String name, int mode)
```

其中，第 1 个参数是文件名称，这个参数不能包含路径，第 2 个参数是操作模式。

Android 系统支持 4 种文件操作模式，如表 7.1 所示。

表 7.1　4 种文件操作模式

模　　式	说　　明
MODE_PRIVATE	私有模式，文件仅能够被创建文件的程序访问，或具有相同 UID 的程序访问
MODE_APPEND	追加模式，如果文件已经存在，则在文件的结尾处添加新数据
MODE_WORLD_READABLE	全局读模式，允许任何程序读取私有文件
MODE_WORLD_WRITEABLE	全局写模式，允许任何程序写入私有文件

使用 openFileOutput()方法创建新文件的代码如下：

```
String FILE_NAME = "fileExample.txt";
FileOutputStream fos = openFileOutput(FILE_NAME,Context.MODE_PRIVATE)
String text = "Some data";
fos.write(text.getBytes());
fos.flush();
fos.close();
```

在上述代码中，首先定义文件的名称为 fileExample.txt，再使用 openFileOutput()方法以私有模式创建文件，调用 write()方法将数据写入文件，调用 flush()方法将缓冲中的数据写入文件，最后调用 close()方法关闭 FileOutputStream。

注意：在调用 close()方法关闭文件前，必须调用 flush()方法，将缓冲区内所有的数据写入文件。如果调用 close()方法前没有调用 flush()，将可能导致部分数据丢失。

2. openFileInput()方法

openFileInput()方法打开一个指定的文件，用于读取数据。openFileInput()方法的格式如下：

```
public FileInputStream openFileInput (String name)
```

其中，第 1 个参数是文件名称，该参数不能包含路径。

使用 openFileInput()方法打开已有文件读取数据的代码如下：

```
String FILE_NAME = "fileExample.txt";
FileInputStream fis = openFileInput(FILE_NAME);
byte[] readBytes = new byte[fis.available()];
while(fis.read(readBytes) != -1){
     }
```

【例 7.2】 写入和读取数据文件举例。

【解题思路】

用户在编辑框中输入数据，当单击“写入”按钮时，数据写入到/data/data/com.application.writeappendreadfile/files/fileWriteAppendRead.txt 文件中，如果用户选择“追加”复选框，数据将会添加到 fileWriteAppendRead.txt 文件的结尾处。当单击“读取”按钮，程序会读取 fileWriteAppendRead.txt 文件的内容，并显示在界面下方的区域内。

【开发步骤和程序分析】

(1) 在 Eclipse 中创建一个 WriteAppendReadFile 应用项目，包名为 com.application.writeappendreadfile。

(2) 设计布局，在 res/layout 目录下的 main.xml 文件中，定义垂直线性布局 LinearLayout，在该布局中，设置一个 TextView 控件和一个 EditText 控件，设置一个内嵌的线性布局，其中分别设置“写入”和“读取”两个按钮，设置“追加”复选框，设置 TextView 控件，用于显示读取文件的内容。在该文件中编辑代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<!-- 定义一个垂直线性布局 LinearLayout -->
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
   <!-- 设置一个 TextView 控件 -->
   <TextView android:id="@+id/label"
       android:layout_width="fill_parent"
       android:layout_height="wrap_content"
       android:text="@string/hello"
   />
```

```
14     <!-- 设置一个 EditText 控件,编辑框名为"输入文件内容"-->
15     <EditText android:id = "@ + id/entry"
16         android:text = "输入文件内容"
17         android:layout_width = "fill_parent"
18         android:layout_height = "wrap_content">
19     </EditText>
20     <!-- 设置一个内嵌的线性布局 -->
21     <LinearLayout android:id = "@ + id/LinearLayout01"
22          android:layout_width = "wrap_content"
23          android:layout_height = "wrap_content">
24          <!-- 设置一个 Button 控件,按钮名为"写入" -->
25          <Button android:id = "@ + id/write"
26              android:text = "写入"
27              android:layout_width = "wrap_content"
28              android:layout_height = "wrap_content">
29          </Button>
30          <!-- 设置一个 Button 控件,按钮名为"读取" -->
31          <Button android:id = "@ + id/read"
32              android:text = "读取"
33              android:layout_width = "wrap_content"
34              android:layout_height = "wrap_content">
35          </Button>
36      </LinearLayout>
37      <!-- 设置一个 CheckBox,复选框名为"追加" -->
38      <CheckBox android:id = "@ + id/append"
39          android:text = "追加"
40          android:layout_width = "wrap_content"
41          android:layout_height = "wrap_content">
42      </CheckBox>
43      <!-- 设置一个 TextView,文本框名为"显示文件内容" -->
44      <TextView android:id = "@ + id/display"
45          android:text = "显示文件内容"
46          android:layout_width = "fill_parent"
47          android:layout_height = "fill_parent"
48          android:background = "#FFFFFF"
49          android:textColor = "#000000" >
50      </TextView>
51 </LinearLayout>
```

① 第 3 行至第 51 行定义一个垂直线性布局。

② 第 15 行至第 19 行设置一个 EditText 控件,用于写入数据。

③ 第 21 行至第 36 行设置一个内嵌的线性布局,在第 25 行至第 29 行和第 31 行至第 35 行分别设置了“写入”和“读取”两个按钮,分别用于将数据写入到文件中和从文件中读取数据。

④ 第 38 行至第 42 行设置“追加”复选框。

⑤ 第 44 行至第 50 行设置 TextView 控件，用于显示读取文件的内容。

(3) 在 com. application. writeappendreadfile 包下的 WriteAppendReadFileActivity. java 文件中，定义 WriteAppendReadFileActivity 类继承 Activity 类，为 writeButton 对象和 readButton 对象的 onClick 事件接口设置监听器，定义 writeButtonListener 对象和 readButtonListener 对象基于监听器接口的事件处理方法。在该文件中编辑代码如下：

```
1   package com.application.writeappendreadfile;
2
3   import java.io.FileInputStream;
4   import java.io.FileNotFoundException;
5   import java.io.FileOutputStream;
6   import java.io.IOException;
7   import com.application.writeappendreadfile.R;
8   import android.app.Activity;
9   import android.content.Context;
10  import android.os.Bundle;
11  import android.view.View;
12  import android.view.View.OnClickListener;
13  import android.widget.Button;
14  import android.widget.CheckBox;
15  import android.widget.EditText;
16  import android.widget.TextView;
17  //定义 WriteAppendReadFileActivity 类继承 Activity 类
18  public class WriteAppendReadFileActivity extends Activity {
19      private final String FILE_NAME = "fileWriteAppendRead.txt";
20      private TextView labelView;
21      private TextView displayView;
22      private CheckBox appendBox ;
23      private EditText entryText;
24
25      @Override
26      public void onCreate(Bundle savedInstanceState) {
27        super.onCreate(savedInstanceState);
28        setContentView(R.layout.main);
29        labelView = (TextView)findViewById(R.id.label);
30        displayView = (TextView)findViewById(R.id.display);
31        appendBox = (CheckBox)findViewById(R.id.append);
32        entryText = (EditText)findViewById(R.id.entry);
33        Button writeButton = (Button)findViewById(R.id.write);
34        Button readButton = (Button)findViewById(R.id.read);
35        //为 writeButton 对象和 readButton 对象的 onClick 事件接口设置监听器
36        writeButton.setOnClickListener(writeButtonListener);
37        readButton.setOnClickListener(readButtonListener);
```

```
        entryText.selectAll();
        entryText.findFocus();
    }

    //定义 writeButtonListener 对象基于监听器接口的事件处理方法
    OnClickListener writeButtonListener = new OnClickListener() {
        @Override
        public void onClick(View v) {
            FileOutputStream fos = null;
            try {
                if (appendBox.isChecked()){
                    fos = openFileOutput(FILE_NAME,Context.MODE_APPEND);
                }
                else {
                    fos = openFileOutput(FILE_NAME,Context.MODE_PRIVATE);
                }
                String text = entryText.getText().toString();
                fos.write(text.getBytes());
                labelView.setText("写入成功,写入长度: " + text.length());
                entryText.setText("");
            } catch (FileNotFoundException e) {
                e.printStackTrace();
            }
            catch (IOException e) {
                e.printStackTrace();
            }
            finally{
                if (fos != null){
                    try {
                        fos.flush();
                        fos.close();
                    } catch (IOException e) {
                        e.printStackTrace();
                    }
                }
            }
        }
    };

    //定义 readButtonListener 对象基于监听器接口的事件处理方法
    OnClickListener readButtonListener = new OnClickListener() {
        @Override
        public void onClick(View v) {
            displayView.setText("");
            FileInputStream fis = null;
```

```
83                try {
84                    fis = openFileInput(FILE_NAME);
85                    if (fis.available() == 0){
86                        return;
87                    }
88                    byte[ ] readBytes = new byte[fis.available()];
89                    while(fis.read(readBytes) != -1){
90                    }
91                    String text = new String(readBytes);
92                    displayView.setText(text);
93                    labelView.setText("文件读取成功,文件长度:" + text.length());
94                } catch (FileNotFoundException e) {
95                    e.printStackTrace();
96              }
97                catch (IOException e) {
98                    e.printStackTrace();
99                }
100           }
101       };
102 }
```

① 第 18 行至第 99 行定义一个类 WriteAppendReadFileActivity 继承 Activity 类。

② 第 35 行和第 36 行分别为 writeButton 对象和 readButton 对象的 onClick 事件接口设置监听器。

③ 第 43 行至第 75 行定义 writeButtonListener 对象基于监听器接口的事件处理方法。第 44 行至第 74 行重写 onClick()方法,第 46 行定义 FileOutputStream 类的对象 fos,第 48 行至第 53 行使用 openFileOutput()方法以私有模式创建文件,第 55 行调用 write()方法将数据写入文件,第 67 行调用 flush()方法将缓冲中的数据写入文件,第 68 行调用 close()方法关闭 FileOutputStream。

④ 第 78 行至第 101 行定义 readButtonListener 对象基于监听器接口的事件处理方法。第 79 行至第 100 行重写 onClick()方法,第 82 行定义 FileInputStream 类的对象 fis,第 84 行使用 openFileInput ()方法创建文件,第 89 行至第 90 行调用 write()方法使用循环从文件读出数据。

【运行结果】

在 Eclipse 中启动模拟器,然后运行项目 WriteAppendReadFile,初始界面如图 7.3 所示,在编辑框中输入数据 Wish you,单击“写入”按钮后的界面如图 7.4 所示。

选择“追加”复选框再输入数据 success!,单击“写入”按钮后的界面如图 7.5 所示,单击“读取”按钮后的界面如图 7.6 所示。

通过 File Explorer 查看/data/data 下的数据,数据写入到/data/data/com.application.writeappendreadfile/files/fileWriteAppendRead.txt 文件中,其位置如图 7.7 所示。



图 7.3　WriteAppendReadFile 项目初始界面



图 7.4　输入数据、单击“写入”按钮后的界面



图 7.5　选择“追加”复选框再输入数据，单击“写入”按钮后的界面



图 7.6　单击“读取”按钮后的界面

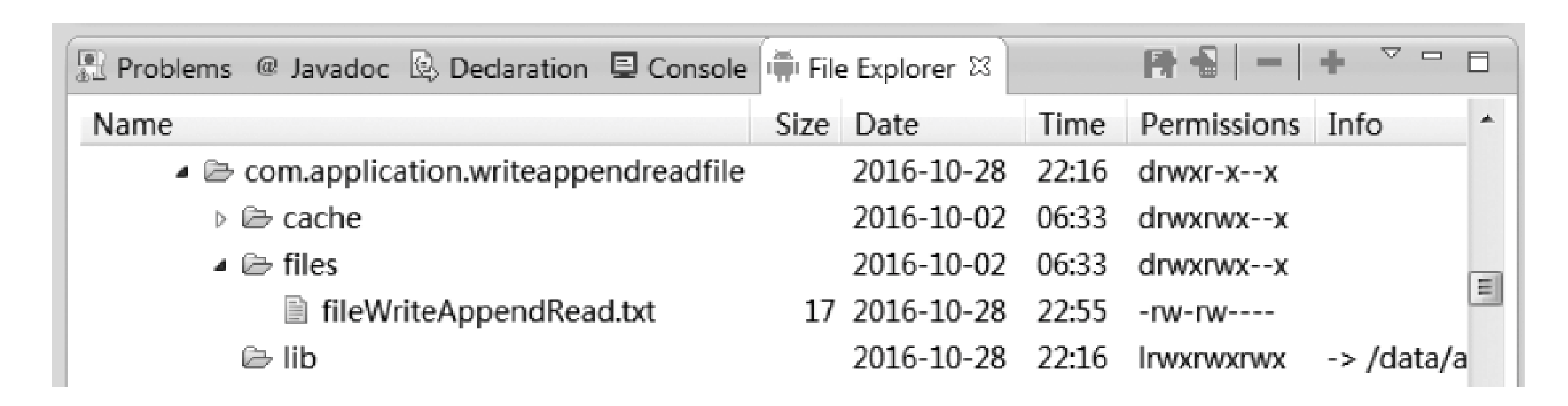


图 7.7 应用程序的数据文件

7.2.2 访问 SD 卡

SD 卡(Secure Digital Memory Card,安全数码卡)是一种广泛使用于数码设备的超小型记忆卡,拥有高记忆容量、移动灵活性、快速数据传输率以及很好的安全性。

Android 模拟器支持 SD 卡的模拟,创建模拟器时可以选择 SD 卡的容量,如图 7.8 所示。

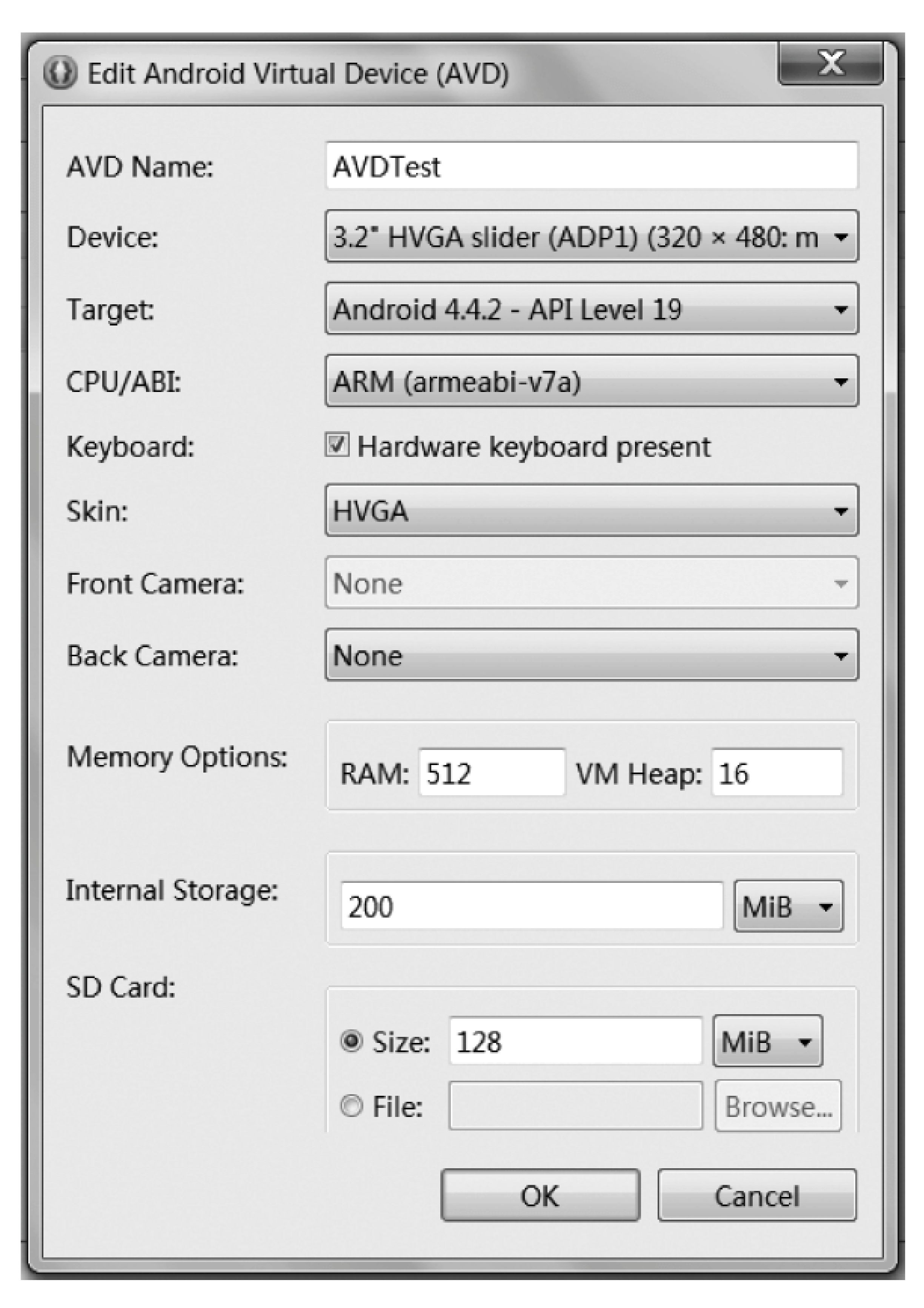


图 7.8 在模拟器中模拟 SD 卡

在模拟器启动时会自动加载 SD 卡,正确加载 SD 卡后,SD 卡中的目录和文件被映射到/mnt/sdcard 目录下。

使用模拟器开发时，可以通过硬盘来模拟 SD 卡，模拟 SD 卡的步骤如下：

(1) 创建一个 SD 卡镜像文件。

(2) 关联 SD 卡和模拟器。

(3) 向 SD 卡中导入文件。

(4) 在模拟器中使用 SD 卡中的文件。

因为用户可以加载或卸载 SD 卡，所以在编程访问 SD 卡前首先需要检测/mnt/sdcard 目录是否可用。

如果不可用，说明设备中的 SD 卡已经被卸载。如果可用，则直接通过使用标准的 java.io.File 类进行访问。

目前 Android 支持 8MB～128GB 的 SD 卡。

【例 7.3】 访问 SD 卡举例，说明如何将数据保存在 SD 卡中。

【解题思路】

通过“生成随机数列”按钮生成 16 个随机小数，再通过“写入 SD 卡”按钮将生成的数据保存在 SD 卡的根目录中，即 Android 系统的/mnt/sdcard 目录中。

【开发步骤和程序分析】

(1) 在 Eclipse 中创建一个 SDcardFileExample 应用项目，包名为 com.application.sdcardfileexample。

(2) 设计布局，在 res/layout 目录下的 main.xml 文件中，定义垂直线性布局 LinearLayout，在该布局中，设置一个 TextView 控件，设置一个内嵌的线性布局，其中分别设置“生成随机数列”和“写入 SD 卡”两个按钮，设置 TextView 控件，用于将生成的随机小数显示在该文本框中。在该文件中编辑代码如下：

```
<?xml version = "1.0" encoding = "utf - 8"?>
<! -- 定义一个垂直线性布局 LinearLayout -->
<LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = "vertical"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    >
    <! -- 设置一个 TextView 控件 -->
    <TextView android:id = "@ + id/label"
            android:layout_width = "fill_parent"
            android:layout_height = "wrap_content"
            android:text = "@string/hello">
    </TextView>
     <! -- 设置一个内嵌的线性布局 -->
     <LinearLayout android:id = "@ + id/LinearLayout01"
          android:layout_width = "wrap_content"
          android:layout_height = "wrap_content">
          <! -- 设置一个 Button 控件,按钮名为"生成随机数列" -->
          <Button android:id = "@ + id/random"
             android:text = "生成随机数列"
```

```
21                android:layout_width = "150dip"
22                android:layout_height = "wrap_content">
23            </Button>
24            <! -- 设置一个 Button 控件,按钮名为"写入 SD 卡" -->
25            <Button android:id = "@ + id/write"
26                android:text = "写入 SD 卡"
27                android:layout_width = "150dip"
28                android:layout_height = "wrap_content">
29            </Button>
30        </LinearLayout>
31          <! -- 设置一个 TextView 控件,用于将生成的随机小数显示在该文本框中 -->
32        <TextView android:id = "@ + id/display"
33            android:layout_width = "fill_parent"
34            android:layout_height = "wrap_content"
35            android:text = "">
36      </TextView>
37 </LinearLayout>
```

① 第 3 行至第 37 行定义一个垂直线性布局。

② 第 15 行至第 30 行设置一个内嵌的线性布局,在第 19 行至第 23 行和第 25 行至第 29 行分别设置了“生成随机数列”和“写入 SD 卡”两个按钮,分别用于生成 16 个随机小数和将生成的数据保存在 SD 卡中。

③ 第 32 行至第 36 行设置一个文本框,用于将生成 16 个随机小数显示在该文本框中。

(3) 在 com.application.sdcardfileexample 包下的 SDcardFileExampleActivity.java 文件中,定义 SDcardFileExampleActivity 类继承 Activity 类,为 randomButton 对象和 writeButton 对象的 onClick 事件接口设置监听器,定义 randomButtonListener 对象和 writeButtonListener 对象基于监听器接口的事件处理方法。在该文件中编辑代码如下:

```
package com.application.sdcardfileexample;

import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import com.application.sdcardfileexample.R;
import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.TextView;
//定义 SDcardFileExampleActivity 类继承 Activity 类
public class SDcardFileExampleActivity extends Activity {
        private static String randomNumbersString = "";

```

```java
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        Button randomButton = (Button)findViewById(R.id.random);
        Button writeButton = (Button)findViewById(R.id.write);
        //为 randomButton 对象和 writeButton 对象的 onClick 事件接口设置监听器
        randomButton.setOnClickListener(randomButtonListener);
        writeButton.setOnClickListener(writeButtonListener);
    }

    //定义 randomButtonListener 对象基于监听器接口的事件处理方法
    OnClickListener randomButtonListener = new OnClickListener() {
        @Override
        public void onClick(View v) {
            randomNumbersString = "";
            for (int i = 0 ; i<16; i++){
                randomNumbersString += Math.random() + "\n";
            }
            TextView displayView = (TextView)findViewById(R.id.display);
            displayView.setText(randomNumbersString);
        }
    };

    //定义 writeButtonListener 对象基于监听器接口的事件处理方法
    OnClickListener writeButtonListener = new OnClickListener() {
        @Override
        public void onClick(View v) {
            String fileName = "SdcardFile-" + System.currentTimeMillis() + ".txt";
            File dir = new File("/sdcard/");
            if (dir.exists() && dir.canWrite()) {
                File newFile = new File(dir.getAbsolutePath() + "/" + fileName);
                FileOutputStream fos  = null;
                        try {
                            newFile.createNewFile();
                            if (newFile.exists() && newFile.canWrite()) {
                            fos  = new FileOutputStream(newFile);
                            fos.write(randomNumbersString.getBytes());
                            TextView labelView = (TextView)findViewById(R.id.label);
                            labelView.setText("随机数列写入 SD 卡");
                        }
                    } catch (IOException e) {
                        e.printStackTrace();
                    } finally {
                  if (fos != null) {
```

```
62                              try {
63                                  fos.flush();
64                                  fos.close();
65                              }
66                              catch (IOException e) { }
67                              }
68                          }
69                      }
70                  }
71              };
72 }
```

① 第 14 行至第 72 行定义一个类 SDcardFileExampleActivity 继承 Activity 类。

② 第 24 行和第 25 行分别为 randomButton 对象和 writeButton 对象的 onClick 事件接口设置监听器。

③ 第 29 行至第 39 行定义 randomButtonListener 对象基于监听器接口的事件处理方法。第 30 行至第 36 行重写 onClick()方法，第 34 行至第 37 行通过循环生成 16 个随机小数并在文本框内显示。

④ 第 42 行至第 71 行定义 writeButtonListener 对象基于监听器接口的事件处理方法。第 43 行至第 70 行重写 onClick()方法，第 45 行为保证在 SD 卡中多次写入时文件名不会重复，在文件名中使用了唯一且不重复的标识，第 47 行在代码中添加了/mnt/sdcard 目录存在性检查，第 48 行使用"绝对目录＋文件名"的形式表示新建立的文件，第 52 行在写入文件前对文件的存在性和可写入性进行检查。

【运行结果】

在 Eclipse 中启动模拟器，然后运行项目 SDcardFileExample，单击"生成随机数列"按钮后，生成的 16 个随机小数显示在下面的文本框中，单击"写入 SD 卡"按钮，生成的数据保存在 SD 卡的根目录中，如图 7.9 所示。



图 7.9　访问 SD 卡界面

7.2.3　访问资源文件

资源文件包括在/res/raw 目录中的原始格式文件和在 res/xml 目录中的 XML 文件。原始格式文件有文本文件(TXT 文件)、数据文件、视频格式文件、音频格式文件、图像文件等。

在应用程序编译和打包时，/res/raw 目录下的所有文件都会保留原有格式不变。而

/res/xml 目录下的 XML 文件会转换为二进制格式，以降低存储器空间占用和提高访问效率。

【例 7.4】 访问资源目录中的数据文件举例。

【解题思路】

当用户单击"读取 TXT 文件"按钮时，程序将读取/res/raw/ts. txt 文件，并将内容显示在界面上，当用户单击"读取 XML 文件"按钮时，程序将读取/res/xml/student. xml 文件，并将内容显示在界面上。

【开发步骤和程序分析】

(1) 在 Eclipse 中创建一个 ResourceFileExample 应用项目，包名为 com. application. resourcefileexample。

(2) 在 res/raw 目录下的 ts. txt 文件中编辑文本如下：

```
存储的数据通常包括文本、声音、图形、动画和活动视频影像.
```

(3) 在/res/xml 目录下的 student. xml 文件中编辑代码如下：

```
<student>
  <person name = "David" sex = "male" totalcredits = "52" />
  <person name = "Mary" sex = "female" totalcredits = "50" />
</student>
```

(4) 设计布局，在 res/layout 目录下的 main. xml 文件中，定义垂直线性布局 LinearLayout，在该布局中，设置一个 TextView 控件，设置一个内嵌的线性布局，其中分别设置"读取 TXT 文件"和"读取 XML 文件"两个按钮，设置一个 TextView 控件，用于显示读取文件的内容，设置一个 TextView，用于清除上述文本框中显示的内容。在该文件中编辑代码如下：

```
<?xml version = "1.0" encoding = "utf - 8"?>
<! -- 定义一个垂直线性布局 LinearLayout -->
<LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = "vertical"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
>
    <! -- 设置一个 TextView 控件 -->
    <TextView  android:id = "@ + id/label"
        android:layout_width = "fill_parent"
        android:layout_height = "wrap_content"
        android:text = "@string/hello">
    </TextView>
     <! -- 设置一个内嵌的线性布局 -->
      <LinearLayout android:id = "@ + id/LinearLayout01"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content">
```

```
        <!-- 设置一个 Button 控件,按钮名为"读取 TXT 文件" -->
        <Button android:id = "@+id/read_raw"
            android:text = "读取 TXT 文件"
            android:layout_width = "150dip"
            android:layout_height = "wrap_content">
        </Button>
        <!-- 设置一个 Button 控件,按钮名为"读取 XML 文件" -->
        <Button android:id = "@+id/read_xml"
            android:text = "读取 XML 文件"
            android:layout_width = "150dip"
            android:layout_height = "wrap_content">
        </Button>
       </LinearLayout>
      <!-- 设置一个 TextView,用于显示读取文件的内容 -->
      <TextView  android:id = "@+id/display"
        android:layout_width = "fill_parent"
        android:layout_height = "wrap_content"
        android:text = "">
     </TextView>
      <!-- 设置一个 TextView,用于清除上述文本框中显示的内容 -->
      <Button android:id = "@+id/clear"
         android:text = "清除显示数据"
         android:layout_width = "150dip"
         android:layout_height = "wrap_content">
     </Button>
</LinearLayout>
```

(5) 在 com.application.resourcefileexample 包下的 ResourceFileExampleActivity.java 文件中，定义 SDcardFileExampleActivity 类继承 Activity 类，为 randomButton 对象和 writeButton 对象的 onClick 事件接口设置监听器，定义 randomButtonListener 对象和 writeButtonListener 对象基于监听器接口的事件处理方法。在该文件中编辑代码如下：

```
package com.application.resourcefileexample;

import java.io.IOException;
import java.io.InputStream;
import org.xmlpull.v1.XmlPullParser;
import com.application.resourcefileexample.R;
import android.app.Activity;
import android.content.res.Resources;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
```

```java
import android.widget.TextView;

public class ResourceFileExampleActivity extends Activity {
    private Resources resources;
    private TextView displayView;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        Button readRawButton = (Button)findViewById(R.id.read_raw);
        Button readXmlButton = (Button)findViewById(R.id.read_xml);
        Button clearButton = (Button)findViewById(R.id.clear);
        //为 readRawButton 对象和 readXmlButton 对象的 onClick 事件接口设置监听器
        readRawButton.setOnClickListener(readRawButtonListener);
        readXmlButton.setOnClickListener(readXmlButtonListener);
        clearButton.setOnClickListener(clearButtonListener);
        this.displayView = (TextView)findViewById(R.id.display);
        this.resources = this.getResources();
    }

    //为 readRawButton 对象和 readXmlButton 对象的 onClick 事件接口设置监听器
    OnClickListener readRawButtonListener = new OnClickListener() {
        @Override
        public void onClick(View v) {
                InputStream inputStream = null;
                try {
                    inputStream = resources.openRawResource(R.raw.ts);
                    byte[] reader = new byte[inputStream.available()];
                    while (inputStream.read(reader) != -1) {
                    }
                    displayView.setText(new String(reader,"utf-8"));
                } catch (IOException e) {
                    Log.e("ResourceFileDemo", e.getMessage(), e);
                } finally {
                    if (inputStream != null) {
                        try {
                            inputStream.close();
                        }
                        catch (IOException e) { }
                    }
                }
        }
    };

```

```
    //定义 readXmlButtonListener 对象基于监听器接口的事件处理方法
    OnClickListener readXmlButtonListener = new OnClickListener() {
        @Override
        public void onClick(View v) {
            XmlPullParser parser = resources.getXml(R.xml.student);
            String msg = "";
            try {
                while (parser.next() != XmlPullParser.END_DOCUMENT) {
                    String student = parser.getName();
                    String name = null;
                    String sex = null;
                    String totalcredits = null;
                    if ((student != null) && student.equals("person")) {
                        int count = parser.getAttributeCount();
                        for (int i = 0; i < count; i++) {
                            String attrName = parser.getAttributeName(i);
                            String attrValue = parser.getAttributeValue(i);
                            if ((attrName != null) && attrName.equals("name")) {
                                name = attrValue;
                            } else if ((attrName != null) && attrName.equals("sex")) {
                                sex = attrValue;
                            } else if ((attrName != null) &&
                              attrName.equals("totalcredits")) {
                                totalcredits = attrValue;
                            }
                        }
                        if ((name != null) && (sex != null) && (totalcredits != null)) {
                        msg += "姓名：" + name + ",性别：" + sex + ",总学分："
                              + totalcredits + "\n";
                        }
                    }
                }
            } catch (Exception e) {
                Log.e("ResourceFileDemo", e.getMessage(), e);
            }
            displayView.setText(msg);
        }
    };

    OnClickListener clearButtonListener = new OnClickListener() {
        @Override
        public void onClick(View v) {
            displayView.setText("");
        }
    };
```

```
104 }
```

① 第 16 行至第 104 行定义一个类 ResourceFileExampleActivity 继承 Activity 类。

② 第 28 行至第 29 行分别 readRawButton 对象和 readXmlButton 对象的 onClick 事件接口设置监听器。

③ 第 36 行至第 57 行定义 readRawButtonListener 对象基于监听器接口的事件处理方法。第 37 行至第 56 行重写 onClick()方法，第 41 行至第 44 行通过调用资源实例的 openRawResource()方法，以二进制流的形式打开指定的 TXT 文件，第 45 行设置显示的文本，使用 UTF-8 的编码方式从字节数组中实例化一个字符串。

④ 第 60 行至第 96 行定义 readXmlButtonListener 对象基于监听器接口的事件处理方法。第 61 行至第 95 行重写 onClick()方法，第 63 行通过资源实例的 getXml()方法获取到 XML 解析器，第 66 行至第 93 行设置循环，通过 parser. next()方法获取解析事件，并通过对比确定事件类型，第 67 行使用 getName()方法获取元素的名称，第 72 行使用 getAttributeCount()方法获取元素的属性数量，第 74 行使用 getAttributeName()方法得到属性名称，第 75 行使用 getAttributeName()方法得到属性值。

【运行结果】

在 Eclipse 中启动模拟器，然后运行项目 ResourceFileExample，初始界面如图 7.10 所示，单击“读取 TXT 文件”按钮后显示文本内容，如图 7.11 所示。





图 7.10　访问资源文件举例的初始界面



图 7.11　读取 TXT 文件

单击“读取 XML 文件”按钮后显示 XML 文件的内容，如图 7.12 所示。



图 7.12　读取 XML 文件

7.3　SQLite 数据库

SQLite 是 Android 自带的轻量级关系数据库，使用资源少，运行高效可靠，可移植性强。

SQLite 嵌入到应用程序内部，数据库、表在内的所有数据都存放在一个单一的文件中，数据库的权限只依赖于文件系统，数据库存储位置在 DDMS 的 File Explorer 中的/data/data/< package name > /databases。

在编程时，一般将数据库的操作都封装在一个类中，调用这个类，就可以完成对数据库的添加、更新、删除和查询等操作。

7.3.1　创建数据库和创建表

1. 创建数据库

在 DBDefinitionManipulation 类中，封装了创建数据库、打开和关闭数据库等操作，下面创建名为 student.db 的数据库，代码如下：

```
public class DBDefinitionManipulation {
    private static final String DB_NAME = "student.db";

    private SQLiteDatabase db;
    private final Context context;
    private DBOpenHelper dbOpenHelper;
```

```
7
8       public DBDefinitionManipulation(Context _context) {
9           context = _context;
10        }
11
12        /** Open the database */
13        public void open() throws SQLiteException {
14            dbOpenHelper = new DBOpenHelper(context, DB_NAME, null, DB_VERSION);
15            try {
16                db = dbOpenHelper.getWritableDatabase();
17            }
18            catch (SQLiteException ex) {
19                db = dbOpenHelper.getReadableDatabase();
20            }
21          }
22
23        public void close() {
24           if (db != null){
25             db.close();
26             db = null;
27                 }
28          }
29 }
```

(1) 第 1 行至第 28 行定义公共类 DBDefinitionManipulation。

(2) 第 2 行声明了数据库的基本信息。

(3) 第 4 行声明了 SQLiteDatabase 类的对象 db，第 6 行声明了 DBOpenHelper 类的对象 dbOpenHelper。

SQLiteDatabase 类封装了较多的方法，用于创建、删除数据库，执行 SQL 命令，对数据进行管理等。

SQLiteOpenHelper 类是一个非常重要的帮助类，这个帮助类可以辅助创建、更新和打开数据库。

(4) 第 13 行至第 21 行使用 open()方法打开数据库，但在 open()方法中没有任何对数据库进行实际操作的代码，而是调用了 SQLiteOpenHelper 类的 getWritableDatabase()方法和 getReadableDatabase()方法。这两个方法会根据数据库是否存在、是否可写等情况，决定在返回数据库实例前，是否需要建立数据库。

(5) 第 23 行至第 28 行在代码第 24 行的 close()方法中，调用了 SQLiteDatabase 对象的 close()方法关闭数据库。

SQLiteDatabase 中也封装了打开数据库的方法 openDatabases()和创建数据库方法 openOrCreateDatabases()，因为代码中使用了帮助类 SQLiteOpenHelper，从而避免直接调用 SQLiteDatabase 的打开和创建数据库的方法，简化了数据库打开过程中烦琐的逻辑判断过程。

2. 创建表

创建一个名为 studentinfo 的表，关系模式为 studentinfo(_id, name, sex, totalcredits)，其中，_id 为学号、整数型、不可空、主键，name 为姓名、可变长字符型、不可空，sex 为性别、可变长字符型、可空，totalcredits 为总学分、整数型、可空。

创建表的代码如下：

```
1  private static class DBOpenHelper extends SQLiteOpenHelper {
2      public DBOpenHelper(Context context, String name, CursorFactory factory, int version) {
3          super(context, name, factory, version);
4      }
5
6      private static final String DB_CREATE = "create table " + studentinfo + " (" + _id
7          + "integer primary key autoincrement," + name + " varchar not null,"
8          + sex + " varchar," + totalcredits + " integer);";
9
10     @Override
11     public void onCreate(SQLiteDatabase _db) {
12         _db.execSQL(DB_CREATE);
13     }
14
15     @Override
16     public void onUpgrade(SQLiteDatabase _db, int _oldVersion, int _newVersion) {
17         _db.execSQL("DROP TABLE IF EXISTS " + studentinfo);
18         onCreate(_db);
19     }
20 }
```

(1) 第 1 行至第 20 行 DBOpenHelper 类继承了帮助类 SQLiteOpenHelper，创建表并重写 onCreate()方法和 onUpgrade()方法。

(2) 第 6 行至第 8 行为创建表的 SQL 语句。

(3) 第 11 行至第 13 行重写 onCreate()方法，这是继承 SQLiteOpenHelper 类必须重写的方法，第 12 行通过调用 SQLiteDatabase 对象的 execSQL()方法，执行创建表的 SQL 命令。

(4) 第 16 行至第 19 行重写了 onUpgrade()方法，onUpgrade()方法在数据库需要升级时被调用，一般用来删除旧的数据库表，并将数据转移到新版本的数据库表中。第 17 行至第 18 行，没有做任何数据转移，仅仅删除原有的表后建立新的表。程序开发人员不应直接调用 onCreate()和 onUpgrade()方法，而应由 SQLiteOpenHelper 类来决定何时调用这两个方法。

7.3.2 数据操纵语句

数据操纵语句指对数据库进行插入、更新、删除、查询等操作。

程序开发人员可以使用 SQL 语句完成以上数据操作，但使用 Android 提供的类和方法则更加简洁、方便。

SQLiteDatabase 类的方法 insert()、update()、delete()和 query()，可执行 SQL 语句对数据库进行插入、更新、删除和查询等操作。

下面分别介绍 SQLiteDatabase 类的 insert()方法、update()方法、delete()方法和 query()方法。

1. 插入语句

插入语句使用 insert()方法，并使用 ContentValues 类的对象，ContentValues 类是一个数据承载容器。

插入语句步骤如下：

(1) 将属性值写入到 ContentValues 对象中。

在创建一个 ContentValues 对象后，调用 ContentValues 对象的 put()方法，将每个属性的值写入到 ContentValues 对象中。

(2) 通过 insert()方法将 ContentValues 对象中的数据写入到数据表中。

使用 SQLiteDatabase 对象的 insert()方法，将 ContentValues 对象中的数据写入到指定的数据表中。

```
1  public long insert(Student student) {
2      ContentValues newValues = new ContentValues();
3
4      newValues.put(name, student.Name);
5      newValues.put(sex, student.Sex);
6      newValues.put(totalcredits, student.Totalcredits);
7
8      return db.insert(studentinfo, null, newValues);
9  }
```

① 第 4 行向 ContentValues 类的对象 newValues 添加一组键-值对，put()方法的第 1 个参数是名称，第 2 个参数是值。

② 第 8 行的 insert()方法中，第 1 个参数是数据表的名称，第 2 个参数是在 NULL 时的替换数据，第 3 个参数是向数据库表中插入的数据。

2. 更新语句

更新语句使用 update()方法，并使用 ContentValues 类的对象。

更新语句步骤如下：

(1) 将属性值写入到 ContentValues 对象中。

在创建一个 ContentValues 对象后，调用 put()方法将属性值写入到 ContentValues 对象中。

(2) 通过 update()方法将 ContentValues 对象中的数据写入到数据表中。

使用 SQLiteDatabase 的 update()方法，并指定数据的更新条件。

```
public long updateOneData(long id , Student student {
    ContentValues updateValues = new ContentValues();
    updateValues.put(name, student.Name);
    updateValues.put(sex, student.Sex);
```

```
5       updateValues.put(totalcredits, people.Totalcredits);
6
7       return db.update(studentinfo, updateValues, _ id + " = " + id, null);
8   }
```

第 7 行 update()方法的第 1 个参数为数据表的名称,第 2 个参数是对数据库表进行更新的数据。第 3 个参数是更新条件。update()方法的返回值表示数据库表中被更新的数据数量。

3. 删除语句

删除数据需要调用当前数据库对象的 delete()方法,并指定表名和删除条件, delete()方法的返回值表示被删除的数据数量。

删除语句有以下两种类型:

(1) 删除表中的所有记录。

在下面的代码中,删除条件为 null,deleteAllData()方法删除表中的所有记录。

```
public long deleteAllData() {
    return db.delete(studentinfo, null, null);
}
```

(2) 删除表中的指定的一条记录。

在下面的代码中,指明需要删除数据的 id 值,deleteOneData()方法仅删除一条记录。

```
public long deleteOneData(long id) {
    return db.delete(studentinfo, _id + " = " + id, null);
}
```

4. 查询语句

在 Android 系统中,数据库查询结果的返回值不是数据集,而是返回数据集的指针,这个指针就是 Cursor 类。

Cursor 类支持在查询结果的数据集中以多种方式移动,并能够获取数据集合的属性名称和序号。

Cursor 类的方法如表 7.2 所示。

表 7.2 Cursor 类的方法

方　　法	说　　明
moveToFirst	将指针移动到第一条数据上
moveToNext	将指针移动到下一条数据上
moveToPrevious	将指针移动到上一条数据上
getCount	获取集合的数据数量
getColumnIndexOrThrow	返回指定属性名称的序号,如果属性不存在则产生异常
getColumnName	返回指定序号的属性名称
getColumnNames	返回属性名称的字符串数组
getColumnIndex	根据属性名称返回序号
moveToPosition	将指针移动到指定的数据上
getPosition	返回当前指针的位置

进行数据查询需要调用 SQLiteDatabase 类的 query()方法，query()方法的语法格式如下：

```
Cursor android.database.sqlite.SQLiteDatabase.query(String table, String[] columns, String
selection, String[] selectionArgs, String groupBy, String having, String orderBy)
```

query()方法的参数说明如下：

- String table：指定表名。
- String[] columns：列名。
- String selection：查询条件。
- String[] selectionArgs：如果使用通配符(?)，则需要在这里指定替换符的具体内容。
- String groupBy：分组方式。
- String having：分组条件。
- String orderBy：排序方式。

查询语句有以下两种类型：

(1) 查询全部数据的代码：

```
public Student[] getAllData() {
Cursor results = db.query(studentinfo, new String[] {_id, name, sex, totalcredits }, null,
null, null, null, null);
return ConvertTo Student(results);
}
```

(2) 根据 id 查询数据的代码：

```
studentinfo(_id, name, sex, totalcredits)
public Student[] getOneData(long id) {
Cursor results = db.query(studentinfo, new String[] {_id, name, sex, totalcredits },
_id + "=" + id, null, null, null, null);
return ConvertTo Student(results);
}
```

【例 7.5】 创建 SQLite 数据库和数据操纵举例。

【解题思路】

使用 SQLite 创建一个名为 student.db 的数据库，在该数据库中创建一个名为 studentinfo 的表，关系模式为 studentinfo(_id, name, sex, totalcredits)。

设置有关按钮可进行数据输入和删除，可按 ID 号查询数据、删除数据和更新数据。

【开发步骤和程序分析】

(1) 在 Eclipse 中创建一个 SQLiteExample 应用项目，包名为 com.application.sqliteexample。

(2) 数据库设计。在该应用项目中，创建 student.db 的数据库，在该数据库中创建一个名为 studentinfo 的表，其表结构如表 7.3 所示。

表 7.3　studentinfo 的表结构

列　　名	数据类型	说　　明
_id	integer	主键，学生信息 ID，自动增加
name	varchar	姓名
sex	varchar	性别
totalcredits	integer	总学分

（3）设计布局。在 res/layout 目录下的 main.xml 文件中，定义垂直线性布局 LinearLayout，在该布局中，嵌入一个相对布局 RelativeLayout，其中分别设置“姓名”“性别”“总学分”文本框和编辑框；嵌入一个线性布局 LinearLayout，其中分别设置“插入数据”按钮、“全部查询”按钮、“清除显示”按钮；嵌入一个相对布局 RelativeLayout，其中设置“ID”文本框和编辑框；嵌入一个线性布局 LinearLayout，其中分别设置“查询”按钮、“更新”按钮、“删除”按钮；设置一个滚动视图 ScrollView。在该文件中编辑代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<!-- 定义一个垂直线性布局 LinearLayout -->
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >

     <!-- 设置一个内嵌的相对布局 RelativeLayout -->
     <RelativeLayout android:id="@+id/RelativeLayout01"
           android:layout_width="wrap_content"
           android:layout_height="wrap_content" >
        <!-- 设置一个 EditText 控件 -->
        <EditText android:id="@+id/name"
            android:text=""
            android:layout_width="280dip"
            android:layout_height="wrap_content"
            android:layout_alignParentRight="true"
            android:layout_marginLeft="10dip" >
        </EditText>
        <!-- 设置一个 TextView 控件，文本框名为"姓名：" -->
        <TextView android:id="@+id/name_label"
            android:text="姓名："
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignParentLeft="true"
            android:layout_toRightOf="@id/name"
            android:layout_alignBaseline="@+id/name">
        </TextView>
        <!-- 设置一个 EditText 控件 -->
```

```
        <EditText android:id="@+id/sex"
            android:text=""
            android:layout_width="280dip"
            android:layout_height="wrap_content"
            android:layout_alignParentRight="true"
            android:layout_marginLeft="10dip"
            android:layout_below="@id/name" >
        </EditText>
        <!-- 设置一个 TextView 控件,文本框名为"性别:" -->
        <TextView android:id="@+id/sex_label"
            android:text="性别:"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignParentLeft="true"
            android:layout_toRightOf="@id/sex"
            android:layout_alignBaseline="@+id/sex" >
        </TextView>
        <!-- 设置一个 EditText 控件 -->
        <EditText android:id="@+id/totalcredits"
            android:layout_width="265dip"
            android:layout_height="wrap_content"
            android:layout_alignParentRight="true"
            android:layout_marginLeft="15dip"
            android:layout_below="@id/sex"
            android:numeric="integer">
        </EditText>
        <!-- 设置一个 TextView 控件,文本框名为"总学分:" -->
        <TextView android:id="@+id/totalcredits_label"
            android:text="总学分:"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignParentLeft="true"
            android:layout_toRightOf="@id/totalcredits"
            android:layout_alignBaseline="@+id/totalcredits">
        </TextView>
    </RelativeLayout>

    <!-- 设置一个内嵌的线性布局 LinearLayout -->
    <LinearLayout android:id="@+id/LinearLayout01"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content">
        <!-- 设置一个 Button 控件,按钮名为"插入数据" -->
        <Button android:id="@+id/add"
            android:text="插入数据"
            android:layout_width="wrap_content"
```

```
                android:layout_height = "wrap_content"
                    android:padding  =  "5dip"
                android:layout_weight = "1">
            </Button>
            <! -- 设置一个 Button 控件,按钮名为"全部查询" -->
            < Button android:id = "@ + id/query_all"
                android:text = "全部查询"
                android:layout_width = "wrap_content"
                android:layout_height = "wrap_content"
                android:padding  =  "5dip"
                android:layout_weight = "1">
            </Button>
            <! -- 设置一个 Button 控件,按钮名为"清除显示" -->
            < Button android:id = "@ + id/clear"
                android:text = "清除显示"
                android:layout_width = "wrap_content"
                android:layout_height = "wrap_content"
                android:padding  =  "5dip"
                android:layout_weight = "1">
            </Button>
        </LinearLayout>

        <! -- 设置一个内嵌的相对布局 RelativeLayout -->
        < RelativeLayout android:id = "@ + id/RelativeLayout02"
          android:layout_width = "wrap_content"
          android:layout_height = "wrap_content" >

           <! -- 设置一个 EditText 控件 -->
           < EditText android:id = "@ + id/id_entry"
                 android:text = ""
                 android:layout_width = "280dip"
                 android:layout_height = "wrap_content"
                 android:layout_alignParentRight = "true"
                 android:layout_marginLeft = "10dip" >
           </EditText>
           <! -- 设置一个 TextView 控件,文本框名为" ID: " -->
           < TextView android:id = "@ + id/id_label"
                 android:text = "ID: "
                 android:layout_width = "wrap_content"
                 android:layout_height = "wrap_content"
                 android:layout_alignParentLeft = "true"
                 android:layout_toRightOf = "@id/id_entry"
                 android:layout_alignBaseline = "@ + id/id_entry">
             </TextView>
         </RelativeLayout>
```

```
        <!-- 设置一个内嵌的线性布局 -->
    <LinearLayout android:id = "@ + id/LinearLayout03"
        android:layout_width = "fill_parent"
        android:layout_height = "wrap_content">
        <!-- 设置一个 Button 控件,按钮名为"查询" -->
        <Button android:id = "@ + id/query"
            android:text = "查询"
            android:layout_width = "50dip"
            android:layout_height = "wrap_content"
            android:padding  =  "3dip"
            android:layout_weight = "1">
        </Button>
        <!-- 设置一个 Button 控件,按钮名为"更新" -->
        <Button android:id = "@ + id/update"
            android:text = "更新"
            android:layout_width = "50dip"
            android:layout_height = "wrap_content"
            android:padding  =  "3dip"
            android:layout_weight = "1">
        </Button>
        <!-- 设置一个 Button 控件,按钮名为"删除" -->
        <Button android:id = "@ + id/delete"
            android:text = "删除"
            android:layout_width = "50dip"
            android:layout_height = "wrap_content"
            android:padding  =  "3dip"
            android:layout_weight = "1">
        </Button>
   </LinearLayout>

<!-- 设置一个 TextView 控件,文本框名为"总学分: " -->
<TextView android:id = "@ + id/label"
        android:text = "查询结果: "
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content">
    </TextView>

        <!-- 设置一个滚动视图 ScrollView -->
    <ScrollView android:layout_width = "fill_parent"
        android:layout_height = "fill_parent">
        <LinearLayout android:layout_width = "fill_parent"
            android:layout_height = "wrap_content"
            android:orientation = "vertical">
            <TextView android:id = "@ + id/display"
```

```
166                     android:text = ""
167                     android:layout_width = "wrap_content"
168                     android:layout_height = "wrap_content">
169                 </TextView>
170             </LinearLayout>
171         </ScrollView>
172 </LinearLayout>
```

① 第 2 行至第 172 行定义一个垂直线性布局 LinearLayout。

② 第 10 行至第 66 行嵌入一个相对布局 RelativeLayout，其中分别设置“姓名”文本框和编辑框、“性别”文本框和编辑框、“总学分”文本框和编辑框。

③ 第 69 行至第 96 行嵌入一个线性布局 LinearLayout，其中分别设置“插入数据”按钮、“全部查询”按钮、“清除显示”按钮。

④ 第 99 行至第 120 行嵌入一个相对布局 RelativeLayout，其中设置“ID”文本框和编辑框。

⑤ 第 123 行至第 150 行嵌入一个线性布局 LinearLayout，其中分别设置“查询”按钮、“更新”按钮、“删除”按钮。

⑥ 第 160 行至第 171 设置一个滚动视图 ScrollView。

(4) 定义 Student 类，其 4 个属性 ID、Name、Sex、Totalcredits 分别对应 studentinfo 表的 4 个列 _id（学生信息 ID）、name（姓名）、sex（性别）、totalcredits（总学分），在 com.application.sqliteexample 包下的 Student.java 文件中，编辑代码如下：

```
public class Student {
    public int ID = -1;
    public String Name;
    public String Sex;
    public int Totalcredits;

    @Override
    public String toString(){
        String result = "";
        result += "ID: " + this.ID + ",";
        result += "姓名: " + this.Name + ",";
        result += "性别: " + this.Sex + ", ";
        result += "总学分: " + this.Totalcredits + ",";
        return result;
    }
}
```

(5) 在 com.application.sqliteexample 包下的 DBDefinitionManipulation.java 文件，用于进行打开和关闭数据库、插入数据、查询数据、删除数据、更新数据等操作，其代码如下：

```
package com.application.sqliteexample;

```

```java
import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteException;
import android.database.sqlite.SQLiteOpenHelper;
import android.database.sqlite.SQLiteDatabase.CursorFactory;

public class DBDefinitionManipulation {
    private static final String DB_NAME = "student.db";
    private static final String DB_TABLE = "studentinfo";
    private static final int DB_VERSION = 1;
    public static final String KEY_ID = "_id";
    public static final String KEY_NAME = "name";
    public static final String KEY_SEX = "sex";
    public static final String KEY_TOTALCREDITS = "totalcredits";
    private SQLiteDatabase db;
    private final Context context;
    private DBOpenHelper dbOpenHelper;

    public DBDefinition Manipulation(Context _context) {
        context = _context;
    }

    //关闭数据库
    public void close() {
        if (db != null){
            db.close();
            db = null;
        }
    }

    //打开数据库
    public void open() throws SQLiteException {
      dbOpenHelper = new DBOpenHelper(context, DB_NAME, null, DB_VERSION);
        try {
            db = dbOpenHelper.getWritableDatabase();
        }
        catch (SQLiteException ex) {
            db = dbOpenHelper.getReadable Database();
        }
    }

    //插入数据
   public long insert(Student student) {
```

```
        ContentValues newValues = new ContentValues();
        //将 Name、Sex、Totalcredits 属性值写入到 ContentValues 对象
        newValues.put(KEY_NAME, student.Name);
        newValues.put(KEY_SEX, student.Sex);
        newValues.put(KEY_TOTALCREDITS, student.Totalcredits);
        //通过 insert()方法将 ContentValues 对象中的数据写入到数据表中
        return db.insert(DB_TABLE, null, newValues);
    }

    //查询全部数据
    public Student[] queryAllData() {
        //通过 Cursor 类返回查询数据集的指针
        Cursor results = db.query(DB_TABLE, new String[] {KEY_ID, KEY_NAME,
              KEY_SEX, KEY_TOTALCREDITS}, null, null, null, null, null);
        return ConvertToStudent(results);
    }

    //查询指定 ID 的数据
    public Student[] queryOneData(long id) {
      //通过 Cursor 类返回查询数据集的指针
      Cursor results = db.query(DB_TABLE, new String[] {KEY_ID, KEY_NAME,
        KEY_SEX, KEY_TOTALCREDITS}, KEY_ID + " = " + id, null, null, null, null);
      return ConvertToStudent(results);
    }

    //显示查询数据
    private Student[] ConvertToStudent(Cursor cursor){
        //获取查询数据集的记录数
        int resultCounts = cursor.getCount();
        if (resultCounts == 0 || !cursor.moveToFirst()){
            return null;
        }
        Student[] students = new Student[resultCounts];
        //遍历查询数据集,分别将数据表 studentinfo 中各列的值赋予 students 数组中
        for (int i = 0; i<resultCounts; i++){
            students[i] = new Student();
            students[i].ID = cursor.getInt(0);
            students[i].Name = cursor.getString(cursor.getColumnIndex(KEY_NAME));
            students[i].Sex = cursor.getString(cursor.getColumnIndex(KEY_SEX));
            students[i].Totalcredits = cursor.getInt(cursor.getColumnIndex(
                KEY_TOTALCREDITS));
            cursor.moveToNext();
        }
        return students;
    }
```

```
93
94     //删除指定 ID 的数据
95     public long deleteOneData(long id) {
96         return db.delete(DB_TABLE, KEY_ID + " = " + id, null);
97     }
98
99     //更新指定 ID 的数据
100    public long updateOneData(long id , Student student){
101        ContentValues updateValues = new ContentValues();
102        //将更新的 Name、Sex、Totalcredits 属性值写入到 ContentValues 对象
103        updateValues.put(KEY_NAME, student.Name);
104        updateValues.put(KEY_SEX, student.Sex);
105        updateValues.put(KEY_TOTALCREDITS, student.Totalcredits);
106        //通过 update()方法将 ContentValues 对象中的数据更新到数据表中
107        return db.update(DB_TABLE, updateValues, KEY_ID + " = " + id, null);
108     }
109
110     //静态 Helper 类,用于建立、更新和打开数据库
111     private static class DBOpenHelper extends SQLiteOpenHelper {
112
113        public DBOpenHelper(Context context, String name, CursorFactory factory,
114            int version) {
115          super(context, name, factory, version);
116        }
117
118        private static final String DB_CREATE = "create table " + DB_TABLE +
119            " (" + KEY_ID + " integer primary key autoincrement, " + KEY_NAME +
120            " text not null, " + KEY_SEX + " integer," + KEY_TOTALCREDITS + " float);";
121
122        @Override
123        public void onCreate(SQLiteDatabase _db) {
124          _db.execSQL(DB_CREATE);
125        }
126
127        @Override
128        public void onUpgrade(SQLiteDatabase _db, int _oldVersion, int _newVersion) {
129          _db.execSQL("DROP TABLE IF EXISTS " + DB_TABLE);
130          onCreate(_db);
131        }
132     }
133 }
```

① 第 11 行至第 138 行定义一个类 DBDefinitionManipulation,用于进行打开和关闭数据库、插入数据、查询数据、删除数据、更新数据等操作。

② 第 46 行至第 55 行将属性值写入到 ContentValues 对象后,通过 insert()方法将

ContentValues 对象中的数据写入到数据表中。

③ 第 58 行至第 63 行查询全部数据，通过 Cursor 类返回查询数据集的指针。

④ 第 66 行至第 71 行查询指定 ID 号的数据，通过 Cursor 类返回查询数据集的指针。

⑤ 第 74 行至第 92 行显示查询数据，第 76 行获取查询数据集的记录数，第 82 行至第 91 行遍历该查询数据集，分别将数据表 studentinfo 中各列的值赋予 students 数组中。

⑥ 第 94 行至第 97 行通过 delete()方法删除指定 ID 号的数据。

⑦ 第 100 行至第 108 行更新指定 ID 号的数据，将更新的属性值写入到 ContentValues 对象后，通过 update()方法将 ContentValues 对象中的数据更新到数据表中。

（6）在 com. application. sqliteexample 包下的 SQLiteExampleActivity. java 文件中，定义 SQLiteExampleActivity 类继承 Activity 类，设置“插入数据”按钮监听，调用 dbOperation. insert()方法插入数据；设置“全部查询”按钮监听，调用 dbOperation. queryAllData()方法查询全部数据；设置“清除显示”按钮监听，调用 displayView. setText()方法清除显示数据；设置“查询”按钮监听，调用 dbOperation. queryOneData()方法查询指定 ID 的数据；设置“删除”按钮监听，调用 dbOperation. deleteOneData()方法删除指定 ID 的数据；设置“更新”按钮监听，调用 dbOperation. updateOneData()方法更新指定 ID 的数据。在该文件中编辑代码如下：

```
package com.application.sqliteexample;

import com.application.sqliteexample.R;
import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
//定义 SQLiteExampleActivity 类继承 Activity 类
public class SQLiteExampleActivity extends Activity {
    private DBDefinitionManipulation dbOperation;
    private EditText nameText;
    private EditText sexText;
    private EditText totalcreditsText;
    private EditText idEntry;
    private TextView labelView;
    private TextView displayView;

    @Override
    public void onCreate(Bundle savedInstanceStaDavidDavidte) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        nameText = (EditText)findViewById(R.id.name);
```

```
        sexText = (EditText)findViewById(R.id.sex);
        totalcreditsText = (EditText)findViewById(R.id.totalcredits);
        idEntry = (EditText)findViewById(R.id.id_entry);
        labelView = (TextView)findViewById(R.id.label);
        displayView = (TextView)findViewById(R.id.display);

        Button addButton = (Button)findViewById(R.id.add);
        Button queryAllButton = (Button)findViewById(R.id.query_all);
        Button clearButton = (Button)findViewById(R.id.clear);
        Button queryButton = (Button)findViewById(R.id.query);
        Button deleteButton = (Button)findViewById(R.id.delete);
        Button updateButton = (Button)findViewById(R.id.update);

        addButton.setOnClickListener(addButtonListener);
        queryAllButton.setOnClickListener(queryAllButtonListener);
        clearButton.setOnClickListener(clearButtonListener);
        queryButton.setOnClickListener(queryButtonListener);
        deleteButton.setOnClickListener(deleteButtonListener);
        updateButton.setOnClickListener(updateButtonListener);
        dbOperation = new DBDefinitionManipulation(this);
        dbOperation.open();
    }

    //设置"插入数据"按钮监听
    OnClickListener addButtonListener = new OnClickListener() {

        @Override
        public void onClick(View v) {
            Student student = new Student();
            //获取通过 EditText 输入的"姓名"、"性别"、"总学分"的数据
            student.Name = nameText.getText().toString();
            student.Sex = sexText.getText().toString();
            student.Totalcredits = Integer.parseInt(totalcreditsText.getText().toString());
            //调用 dbOperation.insert()方法插入数据
            long colunm = dbOperation.insert(student);
            if (colunm == -1 ){
                labelView.setText("添加过程错误!");
            } else {
                labelView.setText("成功添加数据,ID: " + String.valueOf(colunm));
            }
        }
    };

    //设置"全部查询"按钮监听
    OnClickListener queryAllButtonListener = new OnClickListener() {
```

```

        @Override
        public void onClick(View v) {
            //调用 dbOperation.queryAllData()方法查询全部数据
            Student[] students = dbOperation.queryAllData();
            if (students == null){
                labelView.setText("数据库中没有数据");
                return;
            }
            labelView.setText("数据库：");
            String msg = "";
            for (int i = 0 ; i< students.length; i++){
                msg += students[i].toString() + "\n";
            }
            displayView.setText(msg);
        }
    };

    //设置"清除显示"按钮监听
    OnClickListener clearButtonListener = new OnClickListener() {

        @Override
        public void onClick(View v) {
            displayView.setText("");  //调用 displayView.setText("")方法清除显示数据
        }
    };

  //设置"查询"按钮监听
  OnClickListener queryButtonListener = new OnClickListener() {

      @Override
      public void onClick(View v) {
          int id = Integer.parseInt(idEntry.getText().toString());
          //调用 dbOperation.queryOneData(id)方法查询指定 ID 号的数据
          Student[] students = dbOperation.queryOneData(id);
          if (students == null){
              labelView.setText("数据库中没有 ID 为" + String.valueOf(id) + "的数据");
              return;
          }
          labelView.setText("数据库：");
          displayView.setText(students[0].toString());
      }
  };

  //设置"删除"按钮监听
  OnClickListener deleteButtonListener = new OnClickListener() {

```

```
118         @Override
119         public void onClick(View v) {
120             long id = Integer.parseInt(idEntry.getText().toString());
121             //调用 dbOperation.deleteOneData(id)方法删除指定 ID 号的数据
122             long result = dbOperation.deleteOneData(id);
123             String msg = "删除 ID 为" + idEntry.getText().toString() + "的数据" +
124             (result>0?"成功":"失败");
125             labelView.setText(msg);
126         }
127     };
128 
129     //设置"更新"按钮监听
130     OnClickListener updateButtonListener = new OnClickListener() {
131 
132         @Override
133         public void onClick(View v) {
134             Student student = new Student();
135             student.Name = nameText.getText().toString();
136             student.Sex = sexText.getText().toString();
137             student.Totalcredits = Integer.parseInt(totalcreditsText.getText().toString());
138             long id = Integer.parseInt(idEntry.getText().toString());
139             //调用 dbOperation.updateOneData(id, student)方法更新指定 ID 号的数据
140             long count = dbOperation.updateOneData(id, student);
141             if (count == -1 ){
142                 labelView.setText("更新错误!");
143             } else {
144             labelView.setText("更新成功,更新数据" + String.valueOf(count) + "条");
145             }
146         }
147     };
148 }
```

① 第 10 行至第 148 行定义一个类 SQLiteExampleActivity 继承 Activity 类。

② 第 50 行至第 67 行设置“插入数据”按钮监听，在事件监听处理过程中，第 56 行至第 58 行分别获取通过 EditText 输入的“姓名”“性别”“总学分”的数据，第 60 行调用 dbOperation.insert()方法插入数据。

③ 第 70 行至第 87 行设置“全部查询”按钮监听，第 75 行调用 dbOperation.queryAllData()方法查询全部数据。

④ 第 90 行至第 96 行设置“清除显示”按钮监听，第 94 行调用 displayView.setText("")方法清除显示数据。

⑤ 第 99 行至第 113 行设置“查询”按钮监听，第 105 行调用 dbOperation.queryOneData(id)方法查询指定 ID 的数据。

⑥ 第 116 行至第 127 行设置“删除”按钮监听，第 122 行调用 dbOperation.deleteOneData(id)方法删除指定 ID 的数据。

⑦ 第 130 行至第 147 行设置“更新”按钮监听，第 140 行调用 dbOperation.updateOneData(id，student)方法更新指定 ID 的数据。

【运行结果】

在 Eclipse 中启动模拟器，然后运行项目 SQLiteExampl，在 SQLite 数据库中查询全部数据如图 7.13 所示，在 SQLite 数据库中查询 ID 为 3 的数据如图 7.14 所示。



图 7.13　在 SQLite 数据库中查询全部数据



图 7.14　在 SQLite 数据库中查询 ID 为 3 的数据

7.4　数据共享

Android 程序中的数据，例如文件数据和数据库数据，都是私有的。Android 没有公共的内存区域供多个应用共享存储数据，应用程序运行在不同的进程空间中，不同应用程序的数据不能够直接访问。

ContentProvider（数据提供者）支持在多个应用程序中存储和读取数据，是应用程序之间共享数据的一种接口机制，提供一套标准接口用来获取和操作数据。指定需要共享的数据后，其他应用程序可对共享数据进行查询、添加、删除和更新等操作。

7.4.1　ContentProvider

Android 系统中有很多内置数据，例如音频、视频、图像和通讯录等，它们都是通过 ContentProvider 提供给用户使用的。

1. ContentProvider 的调用关系

ContentProvider 可以使用 SQLite 数据库、文件系统、SharedPreferences、网络存储等方法存储数据，ContentProvider 实现底层的数据源有数据库、文件系统或网络等。

继承 ContentProvider 类中的接口方法有添加、删除、查找和更新等，用于实现基本数据操作功能。

每个 ContentProvider 都会对外提供一个公共的 URI，调用者不能直接调用 ContentProvider 的接口方法，需要使用 ContentResolver 对象，通过 URI 间接调用 ContentProvider。

2. URI

URI(Uniform Resource Identifier)即通用资源标识符，用于定位远程或本地的可用资源。

ContentProvider 使用的 URI 语法格式如下：

```
content://<authority>/<data_path>/<id>
```

- content://：通用前缀，表示该 URI 用于 ContentProvider 定位资源。
- <authority>：授权者名称，由类的小写全称组成，用来确定由哪一个 ContentProvider 提供资源。
- <data_path>：数据路径，用于确定请求的是哪个数据集。如果 ContentProvider 提供一个数据集，数据路径则可以省略。如果 ContentProvider 提供多个数据集，则须指明是哪一个数据集。数据路径可以写成多段格式，例如 student/girl 和/student/boy。
- <id>：数据编号，用于唯一确定数据集中的一条记录，匹配数据集中_ID 字段的值。如果请求的数据并不只限于一条数据，则<id>是可以省略的。

例如，请求整个 student 数据集的 URI 应写为：

```
content://com.application.contentproviderexample/student
```

请求 student 数据集中第 6 条数据的 URI 应写为：

```
content://com.application.contentproviderexample/student/6
```

ContentProvider 的数据集类似于数据库的数据表，每行是一条记录，每列具有相同的数据类型。ContentProvider 可以提供多个数据集，调用者使用 URI 对不同数据集的数据进行操作，ContentProvider 数据集如表 7.4 所示。

表 7.4　ContentProvider 数据集

_id	name	sex	totalcredits
1	David	male	52
2	Mary	female	50

3. 创建 ContentProvider 的步骤

继承 ContentProvider 类创建一个新的数据提供者的步骤如下：

1) 继承 ContentProvider

创建的类继承 ContentProvider 后，有以下 6 个方法需要重载：

- delete()：删除数据集。

- insert()：添加数据集。
- qurey()：查询数据集。
- update()：更新数据集。
- onCreate()：初始化底层数据集和建立数据连接等工作。
- getType()：返回指定 URI 的 MIME 数据类型。

如果 URI 是单条数据，则返回的 MIME 数据类型应以 vnd. android. cursor. item 开头。

如果 URI 是多条数据，则返回的 MIME 数据类型应以 vnd. android. cursor. dir/开头。

新建立的类继承 ContentProvider 后，Eclipse 会自动生成需要重写代码框架，下面的代码框架是 Eclipse 自动生成的。

```
import android.content.*;
import android.database.Cursor;
import android.net.Uri;

public class StudentProvider extends ContentProvider{

    @Override
    public int delete(Uri uri, String selection, String[] selectionArgs) {
        //TODO Auto-generated method stub
        return 0;
    }

   @Override
   public String getType(Uri uri) {
       //TODO Auto-generated method stub
       return null;
   }

   @Override
   public Uri insert(Uri uri, ContentValues values) {
       //TODO Auto-generated method stub
       return null;
   }

   @Override
   public boolean onCreate() {
       //TODO Auto-generated method stub
        return false;
   }

   @Override
   public Cursor query(Uri uri, String[] projection, String selection,
         String[] selectionArgs, String sortOrder) {
       //TODO Auto-generated method stub
```

```
        return null;
    }

    @Override
    public int update(Uri uri, ContentValues values, String selection,
            String[] selectionArgs) {
        //TODO Auto - generated method stub
        return 0;
    }
}
```

2) 声明 CONTENT_URI

为了便于使用 URI,可将 URI 的授权者名称和数据路径等内容声明为静态常量,并声明 CONTENT_URI,下面列举声明 CONTENT_URI 的代码:

```
1  public static final String AUTHORITY = "com.application.contentproviderexample ";
2  public static final String PATH_SINGLE = "student/#";
3  public static final String PATH_MULTIPLE = "student";
4  public static final String CONTENT_URI_STRING = "content://" + AUTHORITY + "/" + PATH_MULTIPLE;
5  public static final Uri CONTENT_URI = Uri.parse(CONTENT_URI_STRING);
6  private static final int MULTIPLE_STUDENT = 1;
7  private static final int SINGLE_STUDENT = 2;
8
9  private static final UriMatcher uriMatcher;
10 static{
11         uriMatcher = new UriMatcher(UriMatcher.NO_MATCH);
12         uriMatcher.addURI(AUTHORITY, PATH_SINGLE, MULTIPLE_ STUDENT);
13         uriMatcher.addURI(AUTHORITY, PATH_MULTIPLE, SINGLE_ STUDENT);
14 }
```

其中,第 1 行声明了 URI 的授权者名称,第 2 行声明了单条数据的数据路径,第 3 行声明了多条数据的数据路径,第 4 行声明了 CONTENT_URI 的字符串形式,第 5 行正式声明了 CONTENT_URI,第 6 行声明了多条数据的返回代码,第 7 行声明了单条数据的返回代码,第 9 行声明了 UriMatcher,第 10 行到第 13 行的静态构造方法中,声明了 UriMatcher 的匹配方式和返回代码。

为判断 URI 是单条数据还是多条数据,需要构造一个 UriMatcher,在使用 UriMatcher 时,可以调用 match()方法对指定的 URI 进行判断,列举代码如下:

```
switch(uriMatcher.match(uri)){
    case MULTIPLE_STUDENT:
        //多条数据的处理过程
        break;
    case SINGLE_STUDENT:
        //单条数据的处理过程
        break;
```

```
    default:
        throw new IllegalArgumentException("不支持的 URI:" + uri);
}
```

3）注册 ContentProvider

实现 ContentProvider 类的代码，需要在 AndroidManifest. xml 文件中进行注册，列举代码如下：

```
<application android:icon = "@drawable/icon" android:label = "@string/app_name">
    <provider android:name = "StudentProvider"
        android:authorities = "com.application.contentproviderexample"/>
</application>
```

在上面的注册中，授权者名称为 com. application. contentproviderexample，其实现类是 StudentProvider。

7.4.2 ContentResolver

调用者不能直接调用 ContentProvider 的接口方法，需要使用 ContentResolver 对象，通过 URI 间接调用 ContentProvider。

每个 Android 组件都具有一个 ContentResolver 对象，获取 ContentResolver 对象可以调用 getContentResolver()方法，代码如下：

```
ContentResolver resolver = getContentResolver();
```

1. 查询操作

ContentResolver 使用 query()方法查询 ContentProvider 的数据，例如，查询 ID 为 4 的数据的代码如下：

```
String KEY_ID = "_id";
String KEY_NAME = "name";
String KEY_SEX = "sex";
String KEY_TOTALCREDITS = "totalcredits";

Uri uri = Uri.parse(CONTENT_URI_STRING + "/" + "4");
Cursor cursor = resolver.query(uri, new String[]
    {KEY_ID, KEY_NAME, KEY_SEX, KEY_ TOTALCREDITS}, null, null, null);
```

ContentResolver 的 query()方法语法格式如下：

```
Cursor query(Uri uri, String[] projection, String selection, String[] selectionArgs, String
sortOrder)
```

其中，uri 定义了查询的数据集，projection 定义了返回数据的属性，selection 定义了返回数据的查询条件。

2. 添加操作

ContentResolver 向 ContentProvider 中添加数据有以下两种方法。

1）使用 insert()方法添加一条数据

```
ContentValues values = new ContentValues();
values.put(KEY_NAME, "David");
values.put(KEY_SEX, "male");
values.put(KEY_ TOTALCREDITS, 52);
Uri newUri = resolver.insert(CONTENT_URI, values);
```

2）使用 bultInsert()方法添加多条数据

```
ContentValues[] arrayValues = new ContentValues[8];
//实例化每一个 ContentValues
int count = resolver.bultInsert(CONTENT_URI, arrayValues);
```

3. 删除操作

ContentResolver 删除操作使用 delete()方法。

1）删除单条数据

在 URI 中指定需要删除数据的 ID，例如，删除 ID 为 2 的数据的代码如下：

```
Uri uri = Uri.parse(CONTENT_URI_STRING + "/" + "2");
int result = resolver.delete(uri, null, null);
```

2）需要删除多条数据，

在 selection 中声明删除条件，例如，使用 selection 语句定义删除条件为 ID 大于 6 的数据的代码如下：

```
String selection = KEY_ID + ">6";
int result = resolver.delete(CONTENT_URI, selection, null);
```

4. 更新操作

ContentResolver 更新操作使用 update()方法，可以在 URI 中指定需要更新数据的 ID，也可以在 selection 中声明更新条件，例如，更新 ID 为 5 的数据的代码如下：

```
ContentValues values = new ContentValues();
values.put(KEY_NAME, "David");
values.put(KEY_SEX, "male");
values.put(KEY_ TOTALCREDITS, 52);
Uri uri = Uri.parse(CONTENT_URI_STRING + "/" + "5");
int result = resolver.update(uri, values, null, null);
```

【例 7.6】 ContentResolver 通过 URI 使用 ContentProvider 提供的数据实现数据共享。

【解题思路】

建立两个应用项目 ContentProviderExample 和 ContentResolverExample，ContentProviderExample 作为数据提供者，ContentResolverExample 作为数据调用者，ContentResolverExample 通过 URI 使用 ContentProviderExample 提供的数据。

ContentProviderExample 是一个无界面的应用项目，提供数据存储功能，供其他应用程

序进行数据交换使用，其底层使用 SQLite 数据库，支持数据的插入、删除、更新和查询等基本操作。

ContentResolverExample 应用项目自身没有数据存储功能，通过 URI 访问 ContentProviderExample，从而共享 ContentProviderExample 提供的数据。

为实现数据共享，两个应用项目都包含一个内容完全相同的文件 Student. java，该文件定义了授权者名称、数据路径、MIME 数据类型、CONTENT_URI 和数据项名称等内容。

【开发步骤和程序分析】

1）数据提供者 ContentProviderExample

（1）在 Eclipse 中创建一个应用项目 ContentProviderExample，包名为 com. application. contentproviderexample。

（2）在 com. application. contentproviderexample 包下的 Student. java 文件中编辑代码如下：

```
package com.application.contentproviderexample;

import android.net.Uri;

public class Student{

//MIME 数据类型
public static final String MIME_DIR_PREFIX = "vnd.android.cursor.dir";
public static final String MIME_ITEM_PREFIX = "vnd.android.cursor.item";
public static final String MINE_ITEM = "vnd.application.student";
blic static final String MINE_TYPE_SINGLE = MIME_ITEM_PREFIX + "/" + MINE_ITEM;
public static final String MINE_TYPE_MULTIPLE = MIME_DIR_PREFIX + "/" +
    MINE_ITEM;

//授权者名称、数据路径、CONTENT_URI
public static final String AUTHORITY = "com.application.contentproviderexample";
public static final String PATH_SINGLE = "student/#";
public static final String PATH_MULTIPLE = "student";
public static final String CONTENT_URI_STRING = "content://" + AUTHORITY + "/" +
    PATH_MULTIPLE;
public static final Uri  CONTENT_URI = Uri.parse(CONTENT_URI_STRING);

//数据项名称
public static final String KEY_ID = "_id";
public static final String KEY_NAME = "name";
public static final String KEY_SEX = "sex";
public static final String KEY_TOTALCREDITS = "totalcredits";
}
```

（3）在 com. application. contentproviderexample 包下的 StudentProvider. java 文件中，

定义一个类 StudentProvider 继承 ContentProvider 类，重写 ContentProvider 类的 delete()方法，删除数据；重写 ContentProvider 类的 insert()方法，对 studentinfo 表插入数据；重写 ContentProvider 类的 query()方法，通过 Cursor 类返回查询数据集的指针；重写 ContentProvider 类的 update()方法，更新数据。在该文件中编辑代码如下：

```
package com.application.contentproviderexample;

import android.content.ContentProvider;
ort android.content.ContentUris;
import android.content.ContentValues;
import android.content.Context;
import android.content.UriMatcher;
import android.database.Cursor;
import android.database.SQLException;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;
import android.database.sqlite.SQLiteQueryBuilder;
import android.database.sqlite.SQLiteDatabase.CursorFactory;
import android.net.Uri;
//定义 StudentProvider 类继承 ContentProvider 类
public class StudentProvider extends ContentProvider{

    private static final String DB_NAME = "student.db";
    rivate static final String DB_TABLE = "studentinfo";
    private static final int DB_VERSION = 1;
    private SQLiteDatabase db;
    private DBOpenHelper dbOpenHelper;
    private static final int MULTIPLE_STUDENT = 1;
    private static final int SINGLE_STUDENT = 2;
    private static final UriMatcher uriMatcher;

    static {
        uriMatcher = new UriMatcher(UriMatcher.NO_MATCH);
        uriMatcher.addURI(Student.AUTHORITY,Student.PATH_MULTIPLE,
            MULTIPLE_STUDENT);
        uriMatcher.addURI(Student.AUTHORITY, Student.PATH_SINGLE,
            SINGLE_STUDENT);
    }

    @Override
    public String getType(Uri uri) {
        switch(uriMatcher.match(uri)){
            case MULTIPLE_STUDENT:
                return Student.MINE_TYPE_MULTIPLE;
            case SINGLE_STUDENT:
```

```
                return Student.MINE_TYPE_SINGLE;
            default:
                throw new IllegalArgumentException("Unkown uri:" + uri);
        }
    }

    //重写 ContentProvider 类的 delete()方法
    @Override
    public int delete(Uri uri, String selection, String[] selectionArgs) {
        int count = 0;
        switch(uriMatcher.match(uri)){
            case MULTIPLE_STUDENT:     //删除多条数据
                count = db.delete(DB_TABLE, selection, selectionArgs);
                break;
            case SINGLE_STUDENT:       //删除单条数据
                String segment = uri.getPathSegments().get(1);
                count = db.delete(DB_TABLE, Student.KEY_ID + " = " +
                    segment, selectionArgs);
                break;
            default:
                throw new IllegalArgumentException("Unsupported URI:" + uri);
        }
        getContext().getContentResolver().notifyChange(uri, null);
        return count;
    }

    //重写 ContentProvider 类的 insert()方法,对 studentinfo 表插入数据
    @Override
    public Uri insert(Uri uri, ContentValues values) {
        long id = db.insert(DB_TABLE, null, values);
        if ( id > 0 ){
            Uri newUri = ContentUris.withAppendedId(Student.CONTENT_URI, id);
            getContext().getContentResolver().notifyChange(newUri, null);
            return newUri;
        }
        throw new SQLException("Failed to insert row into " + uri);
    }

    @Override
    public boolean onCreate() {
        Context context = getContext();
        dbOpenHelper = new DBOpenHelper(context, DB_NAME, null, DB_VERSION);
        db = dbOpenHelper.getWritableDatabase();
        if (db == null)
            return false;
```

```
            else
                return true;
        }

        //重写 ContentProvider 类的 query()方法
        @Override
        public Cursor query(Uri uri, String[] projection, String selection,
                String[] selectionArgs, String sortOrder) {
            SQLiteQueryBuilder qb = new SQLiteQueryBuilder();
            qb.setTables(DB_TABLE);
            switch(uriMatcher.match(uri)){
                case SINGLE_STUDENT:
                    qb.appendWhere(Student.KEY_ID + " = " + uri.getPathSegments().get(1));
                    break;
                default:
                break;
            }
            Cursor cursor = qb.query(db, projection, selection, selectionArgs, null, null,
                sortOrder);
            cursor.setNotificationUri(getContext().getContentResolver(), uri);
            return cursor;
        }

        //重写 ContentProvider 类的 update() 方法
        @Override
        public int update(Uri uri, ContentValues values, String selection, String[] selectionArgs) {
            int count;
            switch(uriMatcher.match(uri)){
                case MULTIPLE_STUDENT:        //更新多条数据
                    count = db.update(DB_TABLE, values, selection, selectionArgs);
                    break;
                case SINGLE_STUDENT:          //更新单条数据
                    String segment = uri.getPathSegments().get(1);
                    count = db.update(DB_TABLE, values, Student.KEY_ID + " = " +
                            segment, selectionArgs);
                    break;
                default:
                    throw new IllegalArgumentException("Unknow URI:" + uri);
            }
            getContext().getContentResolver().notifyChange(uri, null);
            return count;
        }

        private static class DBOpenHelper extends SQLiteOpenHelper {

```

```
131         public DBOpenHelper(Context context, String name, CursorFactory factory,
132                 int version) {
133             super(context, name, factory, version);
134         }
135 
136         private static final String DB_CREATE = "create table " +
137             DB_TABLE + " (" + Student.KEY_ID + " integer primary key autoincrement, " +
138             Student.KEY_NAME + " text not null, " + Student.KEY_SEX + " integer," +
139             Student.KEY_TOTALCREDITS + " float);";
140 
141         @Override
142         public void onCreate(SQLiteDatabase _db) {
143             _db.execSQL(DB_CREATE);
144         }
145 
146         @Override
147         public void onUpgrade(SQLiteDatabase _db, int _oldVersion, int _newVersion) {
148             _db.execSQL("DROP TABLE IF EXISTS " + DB_TABLE);
149             onCreate(_db);
150         }
151     }
152 }
```

① 第 16 行至第 152 行定义一个类 StudentProvider 继承 ContentProvider 类，用于重写 ContentProvider 类的 delete()方法、insert()方法、query()方法、update()方法、onCreate()方法、getType()方法等。

② 第 48 行至第 65 行重写 ContentProvider 类的 delete()方法，其中，第 52 行至第 54 行删除多条数据，第 55 行至第 59 行删除单条数据。

③ 第 68 行至第 77 行重写 ContentProvider 类的 insert()方法，其中，第 70 行对 studentinfo 表插入数据。

④ 第 91 行至第 107 行重写 ContentProvider 类的 query()方法，其中，第 103 行至第 106 行通过 Cursor 类返回查询数据集的指针。

⑤ 第 110 行至第 127 行重写 ContentProvider 类的 update()方法，其中，第 114 行至第 116 行更新多条数据，第 117 行至第 121 行更新单条数据。

(4) 编辑在根下的 AndroidManifest.xml 文件，添加相应权限，代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
        package="com.application.contentproviderexample"
        android:versionCode="1"
        android:versionName="1.0" >
    <uses-sdk android:minSdkVersion="14" />
    <application android:icon="@drawable/ic_launcher" android:label="@string/app_name" >
        <provider android:authorities="com.application.contentproviderexample"
```

```
            android:name = "com.application.contentproviderexample.StudentProvider"/>
        </application>
</manifest>
```

2）数据调用者 ContentResolverExample

（1）在 Eclipse 中创建一个应用项目 ContentResolverExample，包名为 com.application.contentresolverexample。

（2）设计布局。ContentResolverExample 应用项目的界面与 SQLiteExample 应用项目的界面基本相同，此处略去其在 res/layout 目录下的 main.xml 文件中的代码。

（3）在 com.application.contentresolverexample 包下的 Student.java 文件，与 ContentProviderExample 应用项目中的 Student.java 文件的文件名称和代码完全相同，请参阅 ContentProviderExample 应用项目的开发步骤和程序分析。

（4）在 com.application.contentresolverexample 包下的 ContentResolverExampleActivity.java 文件中，定义一个类 ContentResolverExampleActivity 继承 Activity 类，设置"插入数据"按钮监听，在事件监听处理过程中，分别获取通过 EditText 输入的"姓名""性别""总学分"的数据，调用 resolver.insert()方法插入数据，设置"全部查询"按钮监听，调用 resolver.queryAllData()方法查询全部数据；设置"清除显示"按钮监听，调用 displayView.setText()方法清除显示数据；设置"查询"按钮监听，调用 resolver.queryOneData(id)方法查询指定 ID 的数据；设置"删除"按钮监听，调用 resolver.deleteOneData(id)方法删除指定 ID 的数据；设置"更新"按钮监听，调用 resolver.updateOneData(id, student)方法更新指定 ID 的数据。在该文件中编辑代码如下：

```
ackage com.application.contentresolverexample;

import com.application.contentresolverexample.R;
import android.app.Activity;
import android.content.ContentResolver;
import android.content.ContentValues;
import android.database.Cursor;
import android.net.Uri;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
//定义一个类 ContentResolverExampleActivity 继承 Activity 类
public class ContentResolverExampleActivity extends Activity {
    private EditText nameText;
    private EditText sexText;
    private EditText totalcreditsText;
    private EditText idEntry;
    private TextView labelView;
```

```
    private TextView displayView;
    private ContentResolver resolver;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        nameText = (EditText)findViewById(R.id.name);
        sexText = (EditText)findViewById(R.id.sex);
        totalcreditsText = (EditText)findViewById(R.id.totalcredits);
        idEntry = (EditText)findViewById(R.id.id_entry);
        labelView = (TextView)findViewById(R.id.label);
        displayView = (TextView)findViewById(R.id.display);

        Button addButton = (Button)findViewById(R.id.add);
        Button queryAllButton = (Button)findViewById(R.id.query_all);
        Button clearButton = (Button)findViewById(R.id.clear);
        Button queryButton = (Button)findViewById(R.id.query);
        Button deleteButton = (Button)findViewById(R.id.delete);
        Button updateButton = (Button)findViewById(R.id.update);

        addButton.setOnClickListener(addButtonListener);
        queryAllButton.setOnClickListener(queryAllButtonListener);
        clearButton.setOnClickListener(clearButtonListener);
        queryButton.setOnClickListener(queryButtonListener);
        deleteButton.setOnClickListener(deleteButtonListener);
        updateButton.setOnClickListener(updateButtonListener);
        resolver = this.getContentResolver();
    }

    //设置"插入数据"按钮监听
    OnClickListener addButtonListener = new OnClickListener() {

    @Override
    public void onClick(View v) {
        ContentValues values = new ContentValues();
        //获取通过EditText输入的"姓名"、"性别"、"总学分"的数据
        values.put(Student.KEY_NAME, nameText.getText().toString());
        values.put(Student.KEY_SEX, sexText.getText().toString());
        values.put(Student.KEY_TOTALCREDITS, Integer.parseInt(totalcreditsText.
            getText().toString()));
        //调用resolver.insert()方法插入数据
        Uri newUri = resolver.insert(Student.CONTENT_URI, values);
        labelView.setText("添加成功,URI:" + newUri);
        }
```

```
    };

    //设置"全部查询"按钮监听
    OnClickListener queryAllButtonListener = new OnClickListener() {

        @Override
        public void onClick(View v) {
            //调用 resolver. query ()方法查询全部数据
            Cursor cursor = resolver.query(Student.CONTENT_URI, new String[] {
                    Student.KEY_ID, Student.KEY_NAME, Student.KEY_SEX,
                    Student.KEY_TOTALCREDITS}, null, null, null);
            if (cursor == null){
                labelView.setText("数据库中没有数据");
                return;
            }
            labelView.setText("数据库: " + String.valueOf(cursor.getCount()) + "条记录");
            String msg = "";
            if (cursor.moveToFirst()){
                do{
                    msg += "ID: " + cursor.getInt(cursor.getColumnIndex(
                            Student.KEY_ID)) + ",";
                    msg += "姓名: " + cursor.getString(cursor.getColumnIndex(
                            Student.KEY_NAME)) + ",";
                    msg += "性别: " + cursor.getString(cursor.getColumnIndex(
                            Student.KEY_SEX)) + ", ";
                    msg += "总学分: " + cursor.getInt(cursor.getColumnIndex(
                            Student.KEY_TOTALCREDITS)) + "\n";
                }while(cursor.moveToNext());
            }
            displayView.setText(msg);
        }
    };

   //设置"清除显示"按钮监听
   OnClickListener clearButtonListener = new OnClickListener() {

       @Override
       public void onClick(View v) {
           //调用 displayView.setText("")方法清除显示数据
           displayView.setText("");
       }
   };

   //设置"查询"按钮监听
   OnClickListener queryButtonListener = new OnClickListener() {
```

```
        @Override
        public void onClick(View v) {
            //调用 resolver. query ()方法查询指定 ID 的数据
            Uri uri = Uri.parse(Student.CONTENT_URI_STRING + "/" +
                    idEntry.getText().toString());
            Cursor cursor = resolver.query(uri, new String[] { Student.KEY_ID,
                    Student.KEY_NAME, Student.KEY_SEX,
                    Student.KEY_TOTALCREDITS}, null, null, null);
            if (cursor == null){
            labelView.setText("数据库中没有数据");
            return;
            }
            String msg = "";
            if (cursor.moveToFirst()){
                msg += "ID: " + cursor.getInt(cursor.getColumnIndex(
                        Student.KEY_ID)) + ",";
                msg += "姓名: " + cursor.getString(cursor.getColumnIndex(
                    Student.KEY_NAME)) + ",";
                msg += "性别: " + cursor.getString(cursor.getColumnIndex(
                        Student.KEY_SEX)) + ", ";
                msg += "总学分: " + cursor.getInt(cursor.getColumnIndex(
                        Student.KEY_TOTALCREDITS)) + "\n";
                }
            labelView.setText("数据库: ");
            displayView.setText(msg);
        }
    };

    //设置"删除"按钮监听
    OnClickListener deleteButtonListener = new OnClickListener() {

        @Override
        public void onClick(View v) {
            //调用 resolver. delete ()方法删除指定 ID 的数据
            Uri uri = Uri.parse(Student.CONTENT_URI_STRING + "/" +
                    idEntry.getText().toString());
            int result = resolver.delete(uri, null, null);
            String msg = "删除 ID 为" + idEntry.getText().toString() + "的数据" +
                    (result > 0?"成功":"失败");
            labelView.setText(msg);
        }
    };

    //设置"更新"按钮监听
```

```
157    OnClickListener updateButtonListener = new OnClickListener() {
158
159        @Override
160        public void onClick(View v) {
161            ContentValues values = new ContentValues();
162            values.put(Student.KEY_NAME, nameText.getText().toString());
163            values.put(Student.KEY_SEX, sexText.getText().toString());
164            values.put(Student.KEY_TOTALCREDITS,
165                    Integer.parseInt(totalcreditsText.getText().toString()));
166            Uri uri = Uri.parse(Student.CONTENT_URI_STRING + "/" +
167                    idEntry.getText().toString());
168            //调用 resolver. update ()方法更新指定 ID 的数据
169            int result = resolver.update(uri, values, null, null);
170            String msg = "更新 ID 为" + idEntry.getText().toString() + "的数据" +
171                    (result > 0?"成功":"失败");
172            labelView.setText(msg);
173        }
174    };
175 }
```

① 第 16 行至第 175 行定义一个类 ContentResolverExampleActivity 继承 Activity 类。

② 第 53 行至第 67 行设置“插入数据”按钮监听，在事件监听处理过程中，第 59 行至第 62 行分别获取通过 EditText 输入的“姓名”“性别”“总学分”的数据，第 64 行调用 resolver. insert()方法插入数据。

③ 第 70 行至第 98 行设置“全部查询”按钮监听，第 75 行至第 77 行调用 resolver. queryAllData()方法查询全部数据。

④ 第 101 行至第 108 行设置“清除显示”按钮监听，第 106 行调用 displayView. setText ("")方法清除显示数据。

⑤ 第 111 行至第 139 行设置“查询”按钮监听，第 116 行至第 120 行调用 resolver. queryOneData(id)方法查询指定 ID 的数据。

⑥ 第 142 行至第 154 行设置“删除”按钮监听，第 147 行至第 149 行调用 resolver. deleteOneData(id)方法删除指定 ID 的数据。

⑦ 第 157 行至第 174 行设置“更新”按钮监听，第 169 行调用 resolver. updateOneData (id, student)方法更新指定 ID 的数据。

(5) 编辑在根下的 AndroidManifest. xml 文件，添加相应权限，代码如下：

```
<?xml version = "1.0" encoding = "utf - 8"?>
< manifest xmlns:android = "http://schemas.android.com/apk/res/android"
    package = "com.application.contentresolverexample"
    android:versionCode = "1"
    android:versionName = "1.0" >
    < uses - sdk android:minSdkVersion = "14" />
    < application
```

```
            android:icon = "@drawable/ic_launcher"
            android:label = "@string/app_name" >
            <activity android:label = "@string/app_name"
                    android:name = "com.application.contentresolverexample.
                    ContentResolverExampleActivity" >
                <intent - filter >
                    <action android:name = "android.intent.action.MAIN" />
                        <category android:name = "android.intent.category.LAUNCHER" />
                </intent - filter >
            </activity >
        </application >
</manifest >
```

【运行结果】

在 Eclipse 中启动模拟器,然后运行项目 ContentProviderExample 和 ContentResolverExample, ContentResolverExample 通过数据共享插入数据如图 7.15 所示,ContentResolverExample 通过数据共享查询全部输入数据如图 7.16 所示。



图 7.15 ContentResolverExample 通过数据共享插入数据



图 7.16 ContentResolverExample 通过数据共享查询全部输入数据

ContentResolverExample 通过数据共享查询 ID 为 2 的数据如图 7.17 所示。



图 7.17　ContentResolverExample 通过数据共享查询 ID 为 2 的数据

7.5　小　　结

本章主要介绍了以下内容：

(1) SharedPreference 提供了一种轻量级的数据存取方法，它是 Android 的一种存储简单信息的机制，其文件保存在/data/data/< package name >/shared_prefs 目录下。

通过 SharedPreferences 开发人员可以键-值对的方式存储，而且 SharedPreferences 完全屏蔽了对文件系统的操作过程，仅通过调用 SharedPreferences 中的方法就可以实现键-值对的保存和读取，SharedPreferences 不仅能够保存数据，还能够实现不同应用程序间的数据共享。

(2) Android 支持标准的 Java I/O 读写方法，可以建立和访问程序自身建立的数据文件，可以将文件保存在 SD 卡等外部存储设备中，也可以访问保存在资源目录中的原始文件和 XML 文件。

Android 系统允许应用程序创建的文件是用于自身访问的私有数据文件，文件保存在内部存储器上，位于 Android 系统的目录：/data/data/< package name >/files。

SD 卡(Secure Digital Memory Card，安全数码卡)是一种广泛使用于数码设备的超小型记忆卡，拥有高记忆容量、移动灵活性、快速数据传输率以及很好的安全性。Android 模拟器支持 SD 卡的模拟，创建模拟器时可以选择 SD 卡的容量。

资源文件包括在/res/raw 目录中的原始格式文件和在 res/xml 目录中的 XML 文件。

原始格式文件有文本文件(TXT 文件)、数据文件、视频格式文件、音频格式文件、图像文件等。

(3) SQLite 是 Android 自带的轻量级关系数据库,使用资源少,运行高效可靠,可移植性强。

SQLite 嵌入到应用程序内部,数据库、表在内的所有数据都存放在一个单一的文件中,数据库的权限只依赖于文件系统,数据库存储位置在 DDMS 的 File Explorer 中的/data/data/< package name >/databases。

(4) SQLiteDatabase 类封装了较多的方法,用于创建、删除数据库和执行插入、更新、删除和查询等功能。

SQLiteDatabase 类的方法 insert()、update()、delete()和 query(),可执行 SQL 语句对数据库进行插入、更新、删除和查询等操作。

SQLiteOpenHelper 类是一个重要的帮助类,这个帮助类可以辅助创建、更新和打开数据库。

(5) ContentProvider(数据提供器)支持在多个应用程序中存储和读取数据,是应用程序之间共享数据的一种接口机制,提供一套标准接口用来获取和操作数据。

URI(Uniform Resource Identifier)即通用资源标识符,用于定位远程或本地的可用资源。

ContentProvider 作为数据提供器,ContentResolver 作为数据调用器,ContentResolver 通过 URI 使用 ContentProvider 提供的数据。

习　题　7

一、选择题

7.1　SharedPreferences 的文件保存的目录是________。

A. /data/data/< package name >/files

B. /data/data/< package name >/shared_prefs

C. /data/data/< package name >/databases

D. /data/shared_prefs

7.2　SQLite 数据库中所有的信息都包含在一个文件夹内,方便管理和维护,体现了 SQLite 数据库的________。

A. 独立性　　B. 跨平台性　　C. 安全性　　D. 隔离性

7.3　SQLite 数据库存储的目录是________。

A. /data/data/< package name >/files

B. /data/data/< package name >/shared_prefs

C. /data/data/< package name >/databases

D. /mnt/files

7.4　在 SQLite 数据库中,使用 insert()方法插入数据,需要首先创建________类的对象。

A. Context　　B. ContentValues　　C. Cursor　　D. Bundle

7.5　在 Android 系统中，数据库查询结果返回的数据集指针是________类。

A. ContentValues　　B. Context　　C. Cursor　　D. Bundle

二、填空题

7.6　SharedPreferences 可以通过________的方式进行存储。

7.7　SQLite 是 Android 自带的________关系数据库。

7.8　SQLiteOpenHelper 类是一个重要的________，它可以辅助创建、更新和打开数据库。

7.9　SQLiteDatabase 类用于________、删除数据库和执行插入、更新、删除和查找等功能。

7.10　SQLiteDatabase 类的方法 update()可执行 SQL 语句对数据库进行________操作。

7.11　ContentProvider 支持在多个应用程序中存储和读取数据，它是应用程序之间________的一种接口机制。

7.12　URI 即通用资源标识符，用于定位远程或本地的________。

7.13　ContentProvider 通过________使用 ContentProvider 提供的数据。

三、问答题

7.14　简述 SharedPreferences 对象的数据读取和数据存入。

7.15　SQLiteDatabase 类有哪些功能？ SQLiteOpenHelper 类有哪些功能？

7.16　简述 SQLite 数据库插入语句、更新语句、删除语句、查询语句的操作步骤。

7.17　简述创建 ContentProvider 的步骤。

7.18　简述创建 ContentResolver 的步骤。

四、应用题

7.19　用户登录界面如图 7.18 所示(参见 4.3.7 节综合实例)，完成用户登录界面的数据存储，当单击“注册”按钮后，提交的注册信息存储到 SharedPreference 中。

7.20　完成用户登录界面的数据存储，当单击“注册”按钮后，提交的注册信息存储到一个数据文件中，用户登录界面如图 7.18 所示。

7.21　完成用户登录界面的数据存储，当单击“注册”按钮后，提交的注册信息存储到 SQLite 数据库的表中，用户登录界面如图 7.18 所示。

7.22　在 SQLite 数据库中，创建 studentmanage 数据库和 course 表(课程表)，course 的表结构如表 7.5 所示。



图 7.18　用户登录界面

表 7.5 course 的表结构

列名	数据类型	说明
_id	integer	主键
cname	varchar	课程名
credit	integer	学分

7.23 在 7.22 题创建的 course 表中，增加插入、更新、删除和查询等功能，并将表 7.6 中的数据插入到 course 表中。

表 7.6 course 的数据

_id	cname	credit
102	数字电路	3
203	数据库系统	3
205	微机原理	4
208	计算机网络	4
801	高等数学	4

7.24 创建一个 ContentProvider，用于共享 7.22 题创建的数据库。

第 8 章　多媒体服务

本章要点

- Android 在 android.graphics 包内提供了 2D 图形库，常用类有 Color 类、Paint 类、Canvas 类、Path 类等。
- OpenGL ES 提供两类绘制 3D 图形的方法：glDrawArrays( ) 方法和 glDrawElements()方法。
- 逐帧动画通过连续地播放一系列的图片文件形成动画，它的三个要素是多幅图片、播放顺序和持续时间。
- 补间动画通过一系列的指令，将一个 View 对象进行位置、尺寸、旋转、透明度等变换从而形成动画，View 对象可以是图片、文本、按钮等对象。
- Android 播放音频的方式有两种：使用 MediaPlayer 类和使用 SoundPool 类。
- 可以使用 MediaPlayer 和 SurfaceView 播放视频，也可以使用 VideoView 播放视频。
- 声音采集通过 MediaRecorder 类来实现。
- 图像采集有两种方法：使用手机自带的 Camera 应用程序和使用 Camera 类获得摄像头信息。

随着 Android 应用的发展，手机已不再是单一的通信工具，而是集成图形制作、动画制作、音频录制和播放、视频录制和播放、照相、娱乐、游戏于一体的智能设备。本章介绍绘制 2D 图形、绘制 3D 图形、制作动画、音频播放与视频播放、声音采集与图像采集等内容。

8.1　绘制 2D 图形

在 Android 中需要通过 Graphics 类来显示 2D 图形，本节介绍 2D 图形绘图类、绘制几何图形、绘制路径、绘制文本等内容。

8.1.1　2D 图形绘图类

Android 在 android.graphics 包内提供了 2D 图形库，常用类有 Color 类、Paint 类、Canvas 类、Path 类等，下面分别介绍。

1. Color 类

Android 中的颜色用 4 个数字表示，分别指定透明度、红色、绿色、蓝色(Alpha、Red、Green、Blue，ARGB)，每个数字取值为 0～255(十进制)，用二进制表示则占 8 位，4 个数字

共占 32 位，为了编程方便，通常采用十六进制，4 个数字共占 8 位。

透明度取 0 时表示完全透明，取 255 时表示颜色完全不透明。红色、绿色、蓝色的数字代表颜色深浅度，数字越小颜色越深，数字越大颜色越浅。

Android 中常用的 12 种颜色常量如表 8.1 所示。

表 8.1 Color 类的 12 种颜色常量

颜色常量	说明	颜色常量	说明
Color. BLACK	黑色	Color. GREEN	绿色
Color. BLUE	蓝色	Color. LTGRAY	浅灰色
Color. CYAN	青绿色	Color. MAGENTA	红紫色
Color. DKGRAY	灰黑色	Color. RED	红色
Color. YELLOW	黄色	Color. TRANSPARENT	透明
Color. GRAY	灰色	Color. WHITE	白色

2. Paint 类

Paint 类表示画笔，它是图形库中重要的类之一，用来描述图形的颜色和风格，如颜色、透明度和填充效果等信息。绘图首先需要调整画笔，画笔调整好以后，再将图形绘制到画布上。

使用 Paint 类时，首先需要创建该类的对象，代码如下：

```
Paint paint = new Paint();
```

Paint 类常用方法如表 8.2 所示。

表 8.2 Paint 类的常用方法

方法名称	说明
setARGB(int a,int r,int g ,int b)	设置颜色，分别用于表示透明度、红色、绿色和蓝色值
setColor(int color)	设置颜色，参数 color 可以通过 Color 类提供的颜色常量进行指定，也可以通过 Color. rgb(int red,int green,int blue)方法指定
setAlpha(int a)	设置透明度，值为 0～255 的整数
setAntiAlias(boolwan a)	指定是否使用抗锯齿功能，如果使用，会使绘图速度变慢
setDither(boolean dither)	指定是否使用图像抖动处理，如果使用，会使图像颜色更加平滑和饱满，使图像更加清晰
setShader(Shader shader)	设置渐变，可以使用 LinearGradient(线性渐变)、RadialGradient(径向渐变)或 SweepGradient(角度渐变)
setShadowLayer(float radius, float dx,float dy,int color)	设置阴影，参数 radius 为阴影角度，dx 和 dy 为阴影在 X 轴和 Y 轴上的距离，color 为阴影颜色，如果 radius 参数值为 0 将没有阴影
setStyle(Paint. Style style)	设置填充风格，参数值为 Style. FILL、Style. FILL_AND_STROKE 或 Style. STROKE
setTextAlign(Paint. Align align)	设置文字对齐方式，参数值为 Align. CENTER、Align. LEFT 或 Align. RIGHT
setTextSize(float textSize)	设置字体大小
setFakeBoldText(boolean fbt)	设置是否为粗体文字

3. Canvas 类

Canvas 类表示画布，使用 Canvas 类的各种方法可以在画布上绘制直线、矩形、圆及其他图形。

在 Android 中绘图，需要先创建一个继承自 View 类的视图，并且在该类中重写其 onDraw(Canvas canvas)方法，然后在显示绘图的 Activity 中添加该视图。

Canvas 类的常用方法如表 8.3 所示。

表 8.3 Canvas 类的常用方法

方法名称	说明
drawPoint(float x, float y, Paint paint)	绘制一个像素点
drawPoints(float[] pts, int offset, int count, Paint paint);	绘制多个像素点
drawLine(float startX, float startY, float endX, float endY, Paint paint)	绘制一条直线
void drawLines(float[] pts, int offset, int count, Paint paint)	绘制多条直线
drawCircle(float cx, float cy, float radius, Paint paint)	绘制圆形
drawArc(RectF rectF, float startAngle, float endAngle, boolean useCenter, Paint paint)	绘制弧
drawRect(RectF rect, Paint paint);	绘制矩形
drawOval(RectF oval, Paint paint)	绘制椭圆
drawPath(Path path, Paint paint)	绘制路径
drawText(String text, float x, float y, Paint paint)	绘制文本
drawTextOnPath(String text, Path path, float hOffset, float vOffset, Paint paint)	绘制沿着指定路径的文本

4. Path 类

创建路径需要通过 Path 类实现，该类常用方法如表 8.4 所示。

表 8.4 Path 类的常用方法

方法名称	说明
addArc(RectF oval, float startAngle, float sweepAngle)	添加弧形路径
addCircle(float x, float y, float radius, Path.Direction dir)	添加圆形路径
addOval(RectF oval, Path.Direction dir)	添加椭圆形路径
addRect(RectF rect, Path.Direction dir)	添加矩形路径
addRoundRect(RectF rect, float rx, float ry, Path.Direction dir)	添加圆角矩形路径
moveTo(float x, float y)	设置开始绘制直线的起始点
lineTo(float x, float y)	在 moveTo()方法设置的起始点与该方法指定的结束点之间画一条直线，如果在调用该方法之前没使用 moveTo()方法设置起始点，那么将从(0,0)点开始绘制直线
quadTo(float x1, float y1, float x2, float y2)	用于根据指定的参数绘制一条线段轨迹
close()	闭合路径

8.1.2 绘制图形

绘制图形包括绘制像素点、绘制直线、绘制圆形、绘制弧、绘制矩形、绘制椭圆、绘制路径、绘制文本等内容，下面分别介绍。

1. 绘制像素点

Canvas 类提供的方法可以绘制一个或多个像素点。绘制一个点时需要使用 drawPoint()方法，绘制多个像素点可以使用 drawPoints()方法，其语法格式如下：

```
drawPoint(float x, float y, Paint paint);                      //绘制一个像素点
drawPoints(float[] pts, Paint paint);                          //绘制多个像素点
drawPoints(float[] pts, int offset, int count, Paint paint);   //绘制多个像素点
```

drawPoint()方法的参数说明如下：

- float x，float y：x 表示像素点的横坐标，y 表示像素点的纵坐标。
- Paint paint：paint 为画笔对象。

drawPoints()方法的参数说明如下：

- float[] pts：pts 表示多个像素点的坐标，数组元素必须是偶数个，两个数组元素为一个像素点的坐标。
- int offset：绘制多个像素点需要使用 offset 参数指定取得数组中连续元素的第 1 个元素的位置，即元素偏移量，offset 参数的值从 0 开始，该参数可以从任意一个元素开始取值。
- int count：count 表示要获取的数组元素个数，count 必须为偶数。

2. 绘制直线

调用 Canvas 类的 drawLine()方法绘制一条直线，调用 drawLines()方法绘制多条直线，其语法格式如下：

```
drawLine(float startX, float startY, float endX, float endY, Paint paint);  //绘制一条直线
drawLines(float[] pts, Paint paint);                                        //绘制多条直线
void drawLines(float[] pts, int offset, int count, Paint paint);            //绘制多条直线
```

drawLines()方法的参数说明如下：

- float startX：startX 为直线开始端点的横坐标。
- float startY：startY 为直线开始端点的纵坐标。
- float endX：endX 为直线结束端点的横坐标。
- float endY：endY 为直线结束端点的纵坐标。
- Paint paint：paint 为画笔对象。

例如，绘制一条直线的代码如下：

```
Paint paint = new Paint();                      //创建画笔
paint.setColor(Color.BLUE);                     //设置画笔颜色为蓝色
paint.setStrokeWidth(2);                        //设置线条宽度
canvas.drawLine(200, 25, 300, 25, paint);       //绘制一条直线
```

drawLines()方法的参数说明如下：

- float[] pts：pts 为绘制多条直线时的端点坐标集合，4 个数组元素（两个为开始端点的横坐标和纵坐标，两个为结束端点的横坐标和纵坐标）为一条直线的坐标。
- int offset：offset 为数组元素的偏移量。
- int count：count 为数组元素的个数，该值是 4 的整倍数。

例如，绘制两条直线的代码如下：

```
Paint paint = new Paint();                          //创建画笔
paint.setColor(Color.MAGENTA);                      //设置画笔颜色为红紫色
paint.setStrokeWidth(2);                            //设置线条宽度
float[] pts = { 20, 55, 150, 35, 200, 55, 300, 55}; //直线端点坐标集合,4 个数组元素为一条
                                                    //直线的坐标
canvas.drawLines(pts, paint);                       //绘制两条直线
```

3. 绘制圆形

绘制圆形使用 Canvas 类的 drawCircle()方法，其语法格式如下：

```
drawCircle(float cx, float cy, float radius, Paint paint);
```

drawCircle()方法的参数说明如下：

- float cx：cx 表示圆心横坐标。
- float cy：cy 表示圆心纵坐标。
- float radius：radius 表示圆半径。
- Paint paint：paint 为画笔对象。

例如，绘制三个圆的代码如下：

```
Paint paint = new Paint();                          //创建画笔
paint.setStrokeWidth(3);                            //设置线条宽度
paint.setAntiAlias(true);                           //使用抗锯齿功能
paint.setStyle(Style.STROKE);                       //设置填充样式为描边
paint.setColor(Color.RED);                          //设置颜色为红色
canvas.drawCircle(225, 110, 30, paint);             //绘制圆形
paint.setColor(Color.BLUE);                         //设置颜色为蓝色
canvas.drawCircle(250, 110, 30, paint);             //绘制圆形
paint.setColor(Color.BLACK);                        //设置颜色为黑色
canvas.drawCircle(275, 110, 30, paint);             //绘制圆形
```

4. 绘制弧

绘制弧使用 Canvas 类的 drawArc()方法，其语法格式如下：

```
drawArc(RectF rectF, float startAngle, float endAngle, boolean useCenter, Paint paint);
```

drawCircle()方法的参数说明如下：

- RectF rectF：rectF 为弧的外切矩形坐标，需要设置该矩形的左上角和右下角的坐标，即 rectF. lef、rectF. top、rectF. right 和 rectF. bottom。

- float startAngle：startAngle 为弧的起始角度。
- float endAngle：endAngle 为弧的结束角度。
- boolean useCenter：useCenter 为 True 时，绘制弧的两个端点连接圆心，useCenter 为 False 时，只绘制弧。

例如，绘制弧的代码如下：

```
Paint paint = new Paint();                                   //创建画笔
paint.setColor(Color.RED);                                   //设置颜色为红色
canvas.drawArc(new RectF(10, 60, 130, 120), 30, 90, true, paint);  //绘制弧，且弧的两个端点
                                                                   //连接圆心
canvas.drawArc(new RectF(40, 60, 160, 140), 30, 90, false, paint); //只绘制弧
```

5. 绘制矩形

调用 Canvas 类的 drawRect()方法绘制普通矩形，调用 Canvas 类的 drawRoundRect()方法绘制圆角矩形，其语法格式如下：

```
drawRect(Rect r, Paint paint);
drawRect(RectF rect, Paint paint);
drawRect(float left, float top, float right ,float bottom, Paint paint);
drawRoundRect(RectF rect, float rx, float ry, Paint paint);
```

drawRect()方法和 drawRoundRect()方法的参数说明如下：

- Rect r：r 表示 Rect 对象。
- RectF rect：rect 表示 RectF 对象。
- float left：left 表示矩形的左边位置。
- float top：top 表示矩形的上边位置。
- float right：right 表示矩形的右边位置。
- float bottom：bottom 表示矩形的下边位置。

例如，绘制圆角矩形的代码如下：

```
Paint paint = new Paint();                                         //创建画笔
paint.setColor(Color.CYAN);                                        //设置颜色为青绿色
paint.setStyle(Style.FILL);                                        //设置填充样式
canvas.drawRoundRect(new RectF(200, 250, 300, 310), 20, 20, paint);  //绘制实心圆角矩形
```

6. 绘制椭圆

调用 Canvas 类的 drawOval()方法绘制椭圆，其语法格式如下：

```
drawOval(RectF oval, Paint paint);
```

drawOval()方法的参数说明如下：

- RectF oval：oval 为 RectF 对象。
- Paint paint：paint 为画笔对象。

例如，绘制椭圆的代码如下：

```
Paint paint = new Paint();                              //创建画笔
paint.setColor(Color.BLUE);                             //设置颜色为蓝色
canvas.drawOval(new RectF(20, 160, 120, 230), paint);   //绘制椭圆
paint.setStyle(Style.FILL);                             //设置填充样式
canvas.drawOval(new RectF(200, 160, 300, 230), paint);  //绘制实心椭圆
```

7. 绘制路径

绘制路径包含两个步骤：创建路径和将定义好的路径绘制在画布上。

1）创建路径

创建路径需要通过 Path 类实现，该类包含一组绘图方法，可参阅表 8.4，在调用其中的 addCircle()、addOval()、addRect()和 addRoundRect()方法时需要指定 Path.Direction 类型的常量，其中，Path.Direction.CW 为顺时针方向，Path.Direction.CCW 为逆时针方向。

例如，创建三角形路径的代码如下：

```
Path path1 = new Path();          //创建路径
path1.moveTo(20, 250);            //设置三角形起始点
path1.lineTo(100, 250);           //设置三角形第 1 条边的结束点，即第 2 条边的起始点
path1.lineTo(70, 310);            //设置三角形第 2 条边的结束点，即第 3 条边的起始点
path1.close();                    //闭合三角形路径
```

例如，创建圆形路径的代码如下：

```
Path path2 = new Path();                              //创建路径
path2.addCircle(250, 120, 50, Path.Direction.CCW);    //设置圆形路径，逆时针方向
path2.close();                                        //闭合圆形路径
```

2）将定义的路径绘制在画布上

使用 Canvas 类的 drawPath()方法将定义好的路径绘制在画布上。

例如，将上述例题定义的 path1 路径和 path2 路径绘制到画布上的代码如下：

```
canvas.drawPath(path1, paint);        //将定义的三角形路径绘制在画布上
canvas.drawPath(path2, paint);        //将定义的圆形路径绘制在画布上
```

8. 绘制文本

绘制文本有两种方法：调用 Canvas 类的 drawText()方法绘制文本，调用 Canvas 类的 drawTextOnPath()方法沿着指定的路径绘制文本。

1）drawText()方法

drawText()方法用于在画布的指定位置绘制文本，其语法格式如下：

```
drawText(String text,float x,float y,Paint paint);
```

drawText()方法的参数说明如下：

- String text：text 表示要绘制的文本。
- float x：x 指定文本的起始横坐标。
- float y：y 指定文本的起始纵坐标。

例如，绘制文本的代码如下：

```
Paint paint2 = new Paint();                     //创建画笔
paint2.setTextSize(25);                         //设置字体大小
paint2.setColor(Color.MAGENTA);                 //设置颜色为紫红色
paint2.setStrokeWidth(3);                       //设置线条宽度
canvas.drawText("身体健康", 40, 360, paint2);     //绘制文本
```

2）drawTextOnPath()方法

drawTextOnPath()方法沿着指定的路径绘制字符串，其语法格式如下：

```
drawTextOnPath(String text, Path path, float hOffset, float vOffset, Paint paint);
drawTextOnPath(char[ ] text, int index, int count, Path path, float hOffset, float vOffset,
Paint paint);
```

drawTextOnPath()方法的参数说明如下：

String text：text 表示绘制的文本。

Path path：path 表示绘制文本时使用的路径。

float hOffset：hOffset 表示绘制文本时相对于路径水平方向的偏移量。

float vOffset：vOffset 表示绘制文本时相对于路径垂直方向的偏移量。

例如，绘制绕圆形路径的环形文本的代码如下：

```
Paint paint = new Paint();                      //创建画笔
paint.setTextSize(30);                          //设置字体大小
paint.setColor(Color.RED);                      //设置颜色为红色
paint.setStrokeWidth(3);                        //设置线条宽度
Path path = new Path();                         //创建路径
path.addCircle(250, 330, 50,Path.Direction.CW); //添加顺时针的圆形路径
paint.setStyle(Style.FILL);                     //设置填充样式
canvas.drawTextOnPath("心情愉快", path, 0, 18, paint);       //绘制绕圆形路径的环形文本
```

8.1.3 绘制 2D 图形举例

为了进一步掌握绘制 2D 图形，下面通过两个例题介绍绘制 2D 图形举例，绘制文字和环形文字。

【例 8.1】 绘制 2D 图形举例。

【解题思路】

绘制 2D 图形举例包括绘制直线、绘制圆形、绘制弧、绘制矩形、绘制椭圆、绘制文本等内容。

【开发步骤和程序分析】

（1）创建项目。在 Eclipse 中创建一个 Draw2DBasis 应用项目，包名为 com. application. draw2dbasis。

（2）设计布局。在 res/layout 目录下的 main. xml 文件中仅定义了一个垂直分布的线性布局，其代码如下：

```
<?xml version = "1.0" encoding = "utf - 8"?>
<LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
```

```
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
</LinearLayout>
```

(3) 编辑代码。在 com. application. draw2dbasis 包下的 Draw2DBasisActivity. java 文件中，定义一个类 Draw2DBasisActivity 继承 Activity 类，定义一个静态内部类 GraphicsView 继承 View 类，重写 onDraw()方法。在该文件中编辑代码如下：

```
package com.application.draw2dbasis;

import com.application.draw2dbasis.R;
import android.app.Activity;
import android.content.Context;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.graphics.Paint.Style;
import android.graphics.Path;
import android.graphics.RectF;
import android.os.Bundle;
import android.view.View;

//定义一个类 Draw2DBasisActivity 继承 Activity 类
public class Draw2DBasisActivity extends Activity {

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(new GraphicsView(this));
    }

    //定义一个静态内部类 GraphicsView 继承 View 类
    static public class GraphicsView extends View {
        public GraphicsView(Context context) {
          super(context);
        }

        //重写 onDraw()方法
        @Override
        protected void onDraw(Canvas canvas) {

            //绘制直线
            Paint paint = new Paint();          //创建画笔
            paint.setColor(Color.RED);          //设置画笔颜色为红色
```

```
paint.setStrokeWidth(2);            //设置线条宽度
canvas.drawLine(20, 30, 150, 30, paint);                //绘制一条直线
paint.setColor(Color.BLUE);         //设置颜色为蓝色
float[] pts = { 20, 50, 150, 50, 200, 50, 300, 30 };
canvas.drawLines(pts, paint);       //绘制两条直线

//绘制圆
paint.setStrokeWidth(3);
paint.setAntiAlias(true);           //设置抗锯齿功能;
paint.setStyle(Style.STROKE);       //设置填充样式为描边
paint.setColor(Color.BLACK);        //设置颜色为黑色
canvas.drawCircle(50, 110, 30, paint);
paint.setColor(Color.RED);          //设置颜色为红色
canvas.drawCircle(75, 110, 30, paint);
paint.setColor(Color.BLUE);         //设置颜色为蓝色
canvas.drawCircle(100, 110, 30, paint);

//绘制弧
paint.setColor(Color.RED);          //设置颜色为红色
canvas.drawArc(new RectF(140, 60, 290, 120), 30, 90, true, paint);
canvas.drawArc(new RectF(170, 60, 320, 140), 30, 90, false, paint);

//绘制椭圆
paint.setColor(Color.MAGENTA);      //设置颜色为紫红色
canvas.drawOval(new RectF(20, 160, 120, 230), paint);
paint.setStyle(Style.FILL);
canvas.drawOval(new RectF(200, 160, 300, 230), paint);

//绘制矩形
paint.setColor(Color.BLUE);         //设置颜色为蓝色
canvas.drawRoundRect(new RectF(200, 250, 300, 310), 20, 20, paint);

//绘制三角形
Path path1 = new Path();            //创建路径
path1.moveTo(20, 250);              //设置起始点
path1.lineTo(100, 250);         //设置第 1 条边的结束点,即第 2 条边的起始点
path1.lineTo(70, 310);          //设置第 2 条边的结束点,即第 3 条边的起始点
path1.close();                      //闭合路径
canvas.drawPath(path1, paint);

//绘制文本
Paint paint2 = new Paint();         //创建画笔
paint2.setTextSize(25);
paint2.setColor(Color.RED);         //设置颜色为红色
paint2.setStrokeWidth(3);           //设置线条宽度
```

```
82                canvas.drawText("春节快乐", 110, 360, paint2);
83            }
84        }
85 }
```

① 第 16 行至第 85 行定义一个类 Draw2DBasisActivity 继承 Activity 类。

② 第 25 行至第 28 行定义一个静态内部类 GraphicsView 继承 View 类。

③ 第 31 行至第 83 行重写 onDraw()方法。

④ 第 35 行至第 41 行绘制直线,第 38 行使用 canvas. drawLine()方法绘制一条直线,第 41 行使用 canvas. drawLines()方法绘制两条直线。

⑤ 第 44 行至第 52 行绘制圆,第 47 行至第 52 行分别使用 canvas. drawCircle()方法绘制不同位置和不同颜色的三个圆。

⑥ 第 55 行至第 57 行绘制弧,第 56 行使用 canvas. drawArc()方法绘制一条两个端点连接圆心的弧,第 57 行使用 canvas. drawArc()方法只绘制一条弧。

⑦ 第 60 行至第 63 行绘制椭圆,第 61 行使用 canvas. drawOval()方法绘制一个椭圆,第 62 行至第 63 行设置填充样式并使用 canvas. drawOval()方法绘制一个实心椭圆。

⑧ 第 66 行至第 67 行绘制矩形,第 67 行使用 canvas. drawRoundRect()方法绘制一个圆角实心矩形(已设置填充样式)。

⑨ 第 70 行至第 75 行绘制三角形,第 70 行通过 Path 类创建路径,第 75 行使用 canvas. drawPath()方法绘制一个实心三角形(已设置填充样式)。

⑩ 第 78 行至第 82 行绘制文本,第 82 行使用 canvas. drawText()方法绘制一个文本。

【运行结果】

在 Eclipse 中启动模拟器,然后运行项目 Draw2DBasis,绘制的直线、圆形、弧、矩形、椭圆和文本如图 8.1 所示。

图 8.1　绘制 2D 图形举例

【例 8.2】　绘制文字和环形文字。

【解题思路】

在界面上绘制文字和环形文字。

【开发步骤和程序分析】

(1) 创建项目。在 Eclipse 中创建一个 Draw2DGraphics 应用项目,包名为 com. application. draw2d。

(2) 设计布局。在 res/layout 目录下的 main. xml 文件中定义了一个垂直分布的线性布局,此处略去代码。

(3) 编辑代码。在 com. application. draw2d 包下的 Draw2DActivity. java 文件中,定义一个

类 Draw2DActivity 继承 Activity 类，定义一个静态内部类 GraphicsView 继承 View 类，定义 GraphicsView 类的构造方法，重写 onDraw()方法，创建画笔，设置文本文字大小、颜色和线条宽度，使用 canvas. drawText ()方法绘制一个文本，使用 canvas. drawPath()方法绘制一个圆，使用 canvas. drawTextOnPath ()方法绘制一个环形文本。在该文件中编辑代码如下：

```
package com.application.draw2d;

import com.application.draw2d.R;
import android.app.Activity;
import android.content.Context;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.graphics.Path;
import android.graphics.Path.Direction;
import android.os.Bundle;
import android.view.View;

//定义一个类 Draw2DActivity 继承 Activity 类
public class Draw2DActivity extends Activity {

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(new GraphicsView(this));
    }

    //定义一个静态内部类 GraphicsView 继承 View 类
    static public class GraphicsView extends View {
        private static final String CIRCLETXT = "界面设计实例: 网上购电脑项目界面设计";
        private final Path circle;
        private final Paint circle_Paint;
        private final Paint circletext_Paint;

        //定义 GraphicsView 类的构造方法
        public GraphicsView(Context context) {
            super(context);
            //创建路径,设置圆路径,顺时针方向
            circle = new Path();
            circle.addCircle(160, 200, 118, Direction.CW);
            //创建画笔,设置圆的相关属性: 消锯齿、空心圆、颜色、线粗细
            circle_Paint = new Paint(Paint.ANTI_ALIAS_FLAG);
            circle_Paint.setStyle(Paint.Style.STROKE);
            circle_Paint.setColor(Color.argb(255, 202, 232, 170));
            circle_Paint.setStrokeWidth(6);
```

```
41                //创建画笔,设置圆内环形文本的相关属性: 消锯齿、实心字、颜色、字号
42                circletext_Paint = new Paint(Paint.ANTI_ALIAS_FLAG);
43                circletext_Paint.setStyle(Paint.Style.FILL_AND_STROKE);
44                circletext_Paint.setColor(Color.BLUE);
45                circletext_Paint.setTextSize(20f);
46                setBackgroundResource(R.drawable.background);      //定义渐变背景色
47            }
48
49            //重写 onDraw()方法
50            @Override
51            protected void onDraw(Canvas canvas) {
52                //创建画笔,设置文本文字大小、颜色和线条宽度
53                Paint text_Paint = new Paint();
54                text_Paint.setTextSize(25);
55                text_Paint.setColor(Color.RED);
56                text_Paint.setStrokeWidth(3);
57                canvas.drawText("高级控件和菜单", 70, 50, text_Paint);
58                canvas.drawPath(circle, circle_Paint);
59                canvas.drawTextOnPath(CIRCLETXT, circle, 0, 26, circletext_Paint);
60            }
61        }
62 }
```

① 第 15 行至第 62 行定义一个类 Draw2DActivity 继承 Activity 类。

② 第 24 行至第 61 行定义一个静态内部类 GraphicsView 继承 View 类。

③ 第 31 行至第 47 行定义 GraphicsView 类的构造方法,其中,第 34 行至第 35 行创建路径,设置圆路径为顺时针方向;第 37 行至第 40 行创建画笔,设置圆的相关属性:消锯齿、空心圆、颜色、线粗细;第 42 行至第 45 行创建画笔,设置圆内环形文本的相关属性:消锯齿、实心字、颜色、字号;第 46 行定义渐变背景色。

④ 第 50 行至第 60 行重写 onDraw()方法,第 53 行至第 56 行创建画笔,设置文本文字大小、颜色和线条宽度,第 57 行使用 canvas.drawText()方法绘制一个文本,第 58 行使用 canvas.drawPath()方法绘制一个圆,第 59 行使用 canvas.drawTextOnPath()方法绘制一个环形文本。

【运行结果】

在 Eclipse 中启动模拟器,然后运行项目 Draw2Dgraphic,绘制的文字和环形文字如图 8.2 所示。



图 8.2　绘制文字和环形文字

8.2 绘制 3D 图形

OpenGL 是专业的 3D 图形软件接口标准，Android 系统内置了 OpenGL ES(OpenGL for Embedded Systems)支持，Android 通过使用 OpenGL ES 实现 3D 编程。

8.2.1 绘制 3D 图形的方法和步骤

1. 绘制 3D 图形的基本知识

3D 图形的最基本的单位是顶点(Vertex)，它代表空间中的一个点，由若干点可以构成多边形，再由多边形构成复杂的空间图形。OpenGL ES 支持的基本图形只有点(Point)、线(Line)和三角形(Triangle)，所有 3D 图形都可以通过上述基本图形组合而成。

顶点由 x、y、z 三个坐标值确定，直线由两个顶点索引值表示，三角形由三个顶点索引值表示，颜色由表示色彩的 R、G、B、A 的值确定。

2. 绘制 3D 图形的方法

OpenGL ES 提供两类绘制 3D 图形的方法：glDrawArrays()方法和 glDrawElements()方法。

1) glDrawArrays()方法

glDrawArrays()方法语法格式如下：

```
public abstract void glDrawArrays(int mode, int first, int count)
```

其顶点的顺序由 vertexBuffer 中的顺序指定，使用 VertexBuffer 绘制。

2) glDrawElements()方法

glDrawElements()方法语法格式如下：

```
public abstract void glDrawElements(int mode, int count, int type, Buffer indices)
```

- Buffer indices：顶点的顺序由参数 indices 指定，可以重新定义顶点的顺序。
- int mode：参数 mode 值可取以下常量：GL_POINTS(绘制独立的点)、GL_LINE_STRIP(绘制一条线段)、GL_LINE_LOOP(绘制一条封闭线段，即首尾相连)、GL_LINES(绘制多条线段)、GL_TRIANGLES(绘制多个三角形，且两两不相邻)、GL_TRIANGLE_STRIP(绘制多个三角形，且两两相邻)、GL_TRIANGLE_FAN(以一个点为顶点绘制多个相邻的三角形)等。

3. 绘制 3D 图形的步骤

绘制 3D 图形的步骤如下：

(1) 在当前 Activity 的 onCreate()方法中实例化 GLSurfaceView。

GLSurfaceView 是一个视图，继承至 SurfaceView，它内嵌的 surface 专门负责 OpenGL 渲染，用于构建一个使用 OpenGL ES 进行部分或全部渲染的应用程序。

(2) 自定义 MyRenderer 实现接口 android.opengl.GLSurfaceView.Renderer，并重写以下三个方法：

```
onSurfaceCreated(GL10 gl, EGLConfig config)
```

```
onSurfaceChanged(GL10 gl, int width, int height)
onDrawFrame(GL10 gl)
```

其中，onSurfaceCreated(GL10 gl，EGLConfig config)方法在 surface 创建以后调用，onSurfaceChanged(GL10 gl，int width，int height)方法在 surface 发生改变以后调用，onDrawFrame(GL10 gl)方法是当任何时候调用一个画图方法的时候调用。

GL10 是一个公共接口，它包含 Java 程序语言为 OpenGL 绑定的核心功能。

(3) 给 GLSurfaceView 对象注册一个 Renderer。

(4) 设置当前上下文内容视图。

8.2.2 绘制 3D 图形举例

在绘制 3D 图形举例中，介绍绘制旋转的非透明彩色立方体。

【例 8.3】 绘制一个自动旋转的非透明彩色立方体。

【解题思路】

在本例中，创建一个类继承 Activity 类，在其中的 onCreate()方法中，创建一个 GLSurfaceView 对象，设置绘图模式。另外创建一个类继承 Renderer 类，用于定义 3D 图形的绘制、渲染等。

【开发步骤和程序分析】

(1) 创建项目。在 Eclipse 中创建一个 Draw3DCube 应用项目，包名为 com.application.Draw3DCube。

(2) 设计布局。在 res/layout 目录下的 main.xml 文件中定义了一个垂直分布的线性布局。

(3) 编辑代码 1。开发启动 Activity 的代码，在 com.application.Draw3DCube 包下的 Draw3DCubeActivity.java 文件中，创建一个类 Draw3DCubeActivity 继承 Activity 类，创建一个 GLSurfaceView 对象，用于绘制彩色立方体表面，设置 Renderer 用于定义 3D 图形的绘制、渲染等工作，将创建好的 GLSurfaceView 设置为当前 Activity 的内容视图。在该文件中编辑代码如下：

```
package com.application.Draw3DCube;

import android.app.Activity;
import android.opengl.GLSurfaceView;
import android.os.Bundle;
import android.view.WindowManager;
import android.view.Window;

//创建一个类 Draw3DCubeActivity 继承 Activity 类
public class Draw3DCubeActivity extends Activity {

    GLSurfaceView GLview;

    //当该 Activity 首次创建时调用,重写 onCreate()方法
    @Override
```

```
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        //设置窗体为全屏模式,无标题
        getWindow().addFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN);
        getWindow().requestFeature(Window.FEATURE_NO_TITLE);

        //创建一个 GLSurfaceView 对象,用于绘制彩色立方体表面
        GLSurfaceView GLview = new GLSurfaceView(this);

        //设置 Renderer 用于定义 3D 图形的绘制、渲染等工作
        GLview.setRenderer(new CubeRenderer(this));

        //设置绘制模式为持续绘制
        GLview.setRenderMode(GLSurfaceView.RENDERMODE_CONTINUOUSLY);

        //将创建好的 GLSurfaceView 设置为当前 Activity 的内容视图
        setContentView(GLview);
    }
}
```

(4) 编辑代码 2。开发 3D 效果渲染器代码,在 com.application.Draw3DCube 包下的 CubeRenderer.java 文件中,定义一个类 CubeRenderer 继承 Renderer 类,定义构造方法,其中调用 createBuffers()方法,重写 onSurfaceCreated 方法,重写 onSurfaceChanged()方法,重写 onDrawFrame()方法,定义 createBuffers()方法,创建顶点缓冲、三角形顶点索引缓冲和颜色缓冲,定义本类中使用的变量和数组,并为顶点、顶点颜色和三角形顶点赋初值。在该文件中编辑代码如下:

```
package com.application.Draw3DCube;

import java.nio.ByteBuffer;
import java.nio.ByteOrder;
import javax.microedition.khronos.egl.EGLConfig;
import javax.microedition.khronos.opengles.GL10;
import android.opengl.GLSurfaceView.Renderer;
import android.opengl.GLU;

//定义一个类 CubeRenderer 继承 Renderer 类
public class CubeRenderer implements Renderer {
    public CubeRenderer(Draw3DCubeActivity main) {
        createBuffers();
    }

    //重写抽象方法 onSurfaceCreated(),当 surface 创建时调用
    public void onSurfaceCreated(GL10 gl, EGLConfig config) {
```

```
        gl.glDisable(GL10.GL_DITHER);            //关闭抗抖动
        gl.glClearColor(0.0f, 0.0f, 0.0f, 1.0f); //设置清除颜色缓冲区时用的 RGBA 颜色值
        gl.glEnable(GL10.GL_DEPTH_TEST);         //启用深度测试
        gl.glDepthFunc(GL10.GL_LEQUAL);
        gl.glClearDepthf(1f);
    }

    //重写抽象方法 onSurfaceChanged,当 surface 改变时调用
    public void onSurfaceChanged(GL10 gl, int width, int height) {
        //设置宽度、高度比
        float aspect = (float) width / (float) (height == 0 ? 1 : height);
        //设置 3D 视窗的位置及大小
        gl.glViewport(0, 0, width, height);
        //设置当前矩阵模式为投影矩阵,并将矩阵重置为单位矩阵
        gl.glMatrixMode(GL10.GL_PROJECTION);
        gl.glLoadIdentity();
        GLU.gluPerspective(gl, 45.0f, aspect, 0.1f, 200.0f);
        GLU.gluLookAt(gl, 5f, 5f, 5f, 0f, 0f, 0f, 0, 1, 0);
    }

        //重写抽象方法 onDrawFrame(),用于绘制图形
        public void onDrawFrame(GL10 gl) {
        //清除颜色缓冲
        gl.glClear(GL10.GL_COLOR_BUFFER_BIT | GL10.GL_DEPTH_BUFFER_BIT);

        //设置当前矩阵堆栈为模型堆栈,并重置堆栈,
        //随后的矩阵操作将应用到要绘制的模型上
        gl.glMatrixMode(GL10.GL_MODELVIEW);

        gl.glLoadIdentity();
        gl.glLightfv(GL10.GL_LIGHT0, GL10.GL_POSITION, new float[]{5, 5, 5, 1}, 0);

        //将旋转矩阵应用到当前矩阵堆栈上,即旋转模型
        gl.glRotatef(anglez, 0, 0, 1);
        gl.glRotatef(angley, 0, 1, 0);
        gl.glRotatef(anglex, 1, 0, 0);
        anglex += 0.1; //递增角度值以便每次以不同角度绘制
        angley += 0.2;
        anglez += 0.3;

        //启用顶点数组、法向量、颜色数组
        gl.glEnableClientState(GL10.GL_VERTEX_ARRAY);
        gl.glEnableClientState(GL10.GL_NORMAL_ARRAY);
        gl.glEnableClientState(GL10.GL_COLOR_ARRAY);

```

```
        //设置顶点数组指针为 ByteBuffer 对象 vertices
        //第一个参数为每个顶点包含的数据长度(以第二个参数表示的数据类型为单位)
        gl.glVertexPointer(3, GL10.GL_FLOAT, 0, vertices);
        gl.glColorPointer(4, GL10.GL_FLOAT, 0, colors);

        //绘制 triangles 表示的三角形
        gl.glDrawElements(GL10.GL_TRIANGLES, triangles.remaining(),
                GL10.GL_UNSIGNED_BYTE, triangles);

        //禁用顶点、法向量、颜色数组
        gl.glDisableClientState(GL10.GL_VERTEX_ARRAY);
        gl.glDisableClientState(GL10.GL_COLOR_ARRAY);
    }

    //定义创建缓冲的方法 createBuffers()
    private void createBuffers() {
        //创建顶点缓冲,顶点数组使用 float 类型,每个 float 长 4 个字节
        vertices = ByteBuffer.allocateDirect(data_vertices.length * 4);
        //设置字节顺序为本机顺序
        vertices.order(ByteOrder.nativeOrder());
        //通过一个 FloatBuffer 适配器,将 float 数组写入 ByteBuffer 中
        vertices.asFloatBuffer().put(data_vertices);
        //重置 Buffer 的当前位置
        vertices.position(0);

        //创建三角形顶点索引缓冲,索引使用 byte 类型,所以无须设置字节顺序,
        //也无须写入适配器
        triangles = ByteBuffer.allocateDirect(data_triangles.length * 2);
        triangles.put(data_triangles);
        triangles.position(0);

        colors = ByteBuffer.allocateDirect(data_colors.length * 4);
        colors.order(ByteOrder.nativeOrder());
        colors.asFloatBuffer().put(data_colors);
        colors.position(0);
    }

    private ByteBuffer vertices;
    private ByteBuffer triangles;
    private ByteBuffer colors;

    private float anglex = 0f;
    private float angley = 0f;
    private float anglez = 0f;

```

```
108     //定义立方体的 8 个顶点
109     private float[] data_vertices = {
110         1, 1,  1,  1, -1,  1,   -1, -1,  1,   -1, 1,  1,
111         1, 1, -1,  1, -1, -1,   -1, -1, -1,   -1, 1, -1,
112     };
113
114     //定义立方体的 8 个顶点的颜色
115     private float[] data_colors = {
116         1, 1, 0, 1,  1, 0, 1, 1,  0, 1, 1, 1,  1, 0, 0, 1,
117         0, 0, 0, 1,  0, 0, 1, 1,  0, 1, 0, 1,  1, 1, 1, 1,
118     };
119
120     //定义立方体的 6 个面,共 12 个三角形所需的顶点
121     private byte[] data_triangles = {
122         0, 1, 2,  0, 2, 3,  0, 3, 7,  0, 7, 4,  0, 4, 5,  0, 5, 1,
123         6, 5, 4,  6, 4, 7,  6, 7, 3,  6, 3, 2,  6, 2, 1,  6, 1, 5
124     };
125 }
```

① 第 11 行至第 125 行定义一个类 CubeRenderer 继承 Renderer 类。

② 第 12 行至第 14 行定义构造方法,其中调用 createBuffers()方法。

③ 第 17 行至第 23 行重写 onSurfaceCreated 方法。

④ 第 26 行至第 36 行重写 onSurfaceChanged()方法。

⑤ 第 39 行至第 75 行重写 onDrawFrame()方法,其中,第 41 行至第 48 行重置缓冲、堆栈和光源,第 51 行至第 56 行设置旋转模型,第 59 行至第 61 行启用顶点数组、法向量、颜色数组,为绘制 3D 图形提供数据,第 65 行至第 70 行绘制图形和着色,第 73 行至第 74 行禁用顶点、法向量、颜色数组。

⑥ 第 78 行至第 98 行定义 createBuffers()方法,创建顶点缓冲、三角形顶点索引缓冲和颜色缓冲。

⑦ 第 109 行至第 124 行定义本类中使用的变量和数组,并为顶点、顶点颜色和三角形顶点赋初值。

【运行结果】

在 Eclipse 中启动模拟器,然后运行项目 Draw3Dcube,出现彩色渐变旋转立方体,如图 8.3 所示。

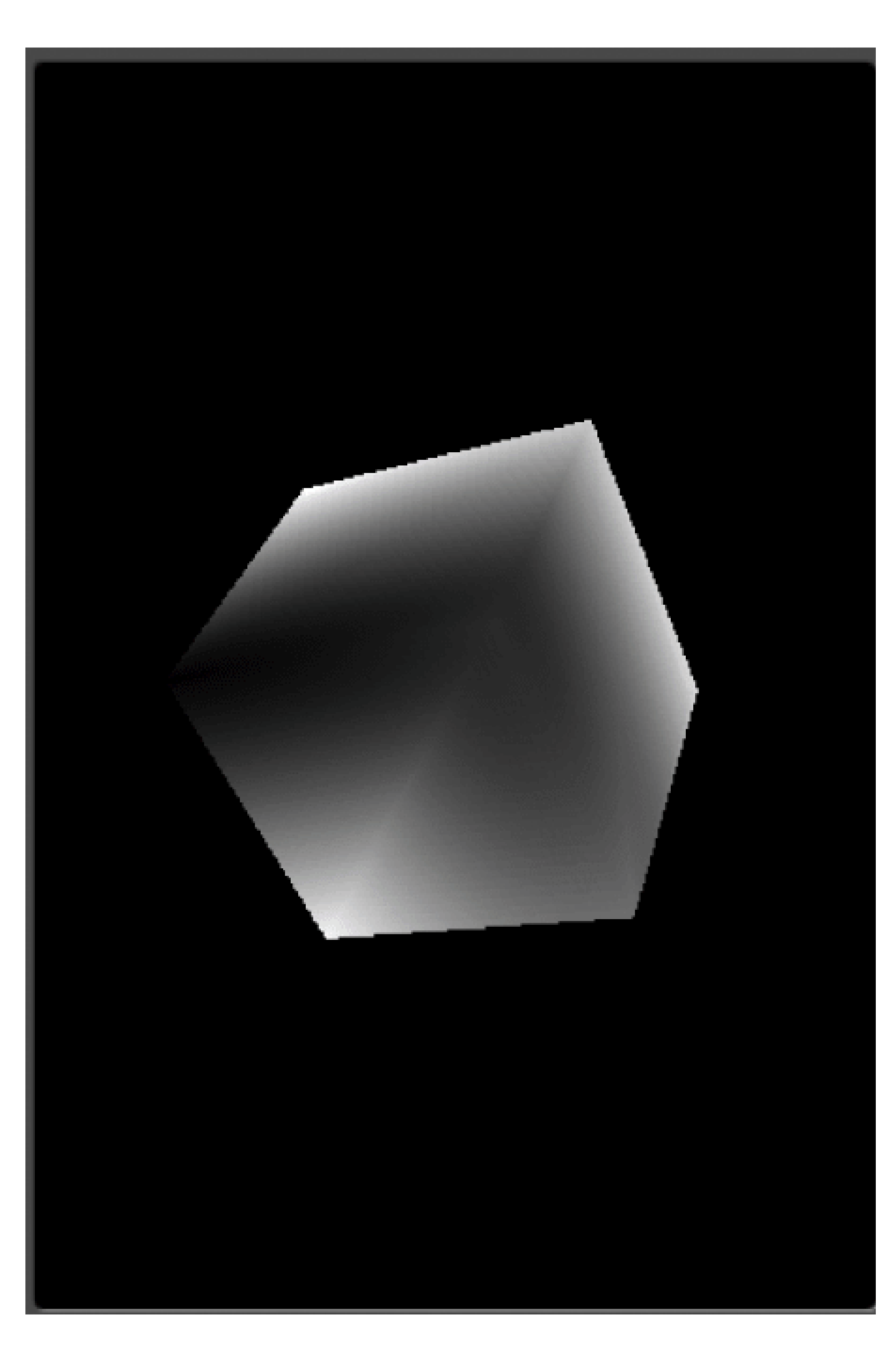


图 8.3 彩色渐变旋转立方体

8.3 制作动画

在 Android 中，制作动画可以使用 Java 代码编程，也可以使用 XML 文件定义动画，其 XML 文件存放在 res/anim 目录下。由于使用 XML 文件定义动画程序开发的效率高，文件的可读性高，本书使用 XML 文件定义动画。

通常将动画分为逐帧动画（Frame Animation）和补间动画（Tween Animation），下面分别介绍。

8.3.1 逐帧动画

逐帧动画通过连续地播放一系列的图片文件，形成动画，它的三个要素是多幅图片、播放顺序和持续时间，逐帧动画用到的类 AnimationDrawable 包含在 android. graphics. drawable. AnimationDrawable 包下，逐帧动画 XML 文件使用的标记和属性如表 8.5 所示。

表 8.5 逐帧动画 XML 文件使用的标记和属性

标记名称	属性值	说明
<animation-list>	android:oneshot：如果设置为 True，则该动画只播放一次，然后停止在最后一帧	Frame Animation 的根标记，包含若干个<item>
<item>	android:drawable：图片帧的引用 android:duration：图片帧的停留时间 android:visible：图片帧是否可见	每个<item>标记定义一个图片帧，其中包含图片资源的引用属性

【例 8.4】 制作逐帧动画举例。

【解题思路】

制作一个循环播放树叶叶片的逐帧动画，可由单击“逐帧动画停止”按钮停止播放，单击“逐帧动画显示”按钮继续播放。

【开发步骤和程序分析】

(1) 创建项目。在 Eclipse 中创建一个 FrameAnimationExample 应用项目，包名为 com. application. frameanimationexample。

(2) 准备图片。准备好一套尺寸相等的树叶叶片的图片，命名为 picture1. png、picture2. png、picture3. png、picture4. png，复制到 res/drawable-mdpi 目录下。

(3) 定义动画。编写动画属性的描述代码，在 res 目录下新建 anim 目录，在该目录中创建 frame. xml 文件，其代码如下：

```
<animation-list xmlns:android="http://schemas.android.com/apk/res/android" android:oneshot="false">
    <item android:drawable="@drawable/picture1" android:duration="500"></item>
    <item android:drawable="@drawable/picture2" android:duration="500"></item>
    <item android:drawable="@drawable/picture3" android:duration="500"></item>
```

```
    <item android:drawable="@drawable/picture4" android:duration="500"></item>
</animation-list>
```

(4) 设计布局。在 res/layout 目录下的 main.xml 文件中，定义垂直线性布局 LinearLayout，在该布局中，设置一个 ImageView 控件，设置一个内嵌的水平线性布局，其中分别设置“逐帧动画显示”和“逐帧动画停止”两个按钮。在该文件中编辑代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<!-- 声明一个垂直分布的线性布局 -->
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >

    <!-- 声明一个 ImageView 控件 -->
    <ImageView
        android:id="@+id/animationIV"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:padding="5px"
        android:src="@anim/frame"
        android:layout_gravity="center_horizontal" />

    <!-- 声明一个内嵌的水平分布线性布局 -->
    <LinearLayout
        android:orientation="horizontal"
        android:paddingTop="25px"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"
    >
        <!-- 声明一个 Button 控件，其 ID 为 buttonA，按钮名为"逐帧动画显示" -->
        <Button
            android:id="@+id/buttonA"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:padding="5px"
            android:text="逐帧动画显示" />
        <!-- 声明一个 Button 控件，其 ID 为 buttonB，按钮名为"逐帧动画停止" -->
        <Button
            android:id="@+id/buttonB"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:padding="5px"
            android:text="逐帧动画停止" />
```

```
39     </LinearLayout>
40 </LinearLayout>
```

第 14 行设置 ImageView 控件的图片源为 anim 目录下的 frame. png 文件。

(5) 编辑代码。在 com. application. frameanimationexample 包下的 FrameAnimationExampleActivity. java 文件中，定义一个类 FrameAnimationExampleActivity 继承 Activity 类，重写 onCreate 方法，分别获取按钮对象 buttonA 和 buttonB；为按钮对象 buttonA 设置监听器，通过 animationDrawable 对象的 start()方法启动逐帧动画；为按钮对象 buttonB 设置监听器，通过 animationDrawable 对象的 stop()方法停止逐帧动画。在该文件中编辑代码如下：

```
package com.application.frameanimationexample;

import com.application.frameanimationexample.R;
import android.app.Activity;
import android.graphics.drawable.AnimationDrawable;
import android.os.Bundle;
import android.view.View;
import android.view.Window;
import android.widget.Button;
import android.widget.ImageView;
import android.view.View.OnClickListener;

//定义一个类 FrameAnimationExampleActivity 继承 Activity 类
public class FrameAnimationExampleActivity extends Activity {
    private ImageView animationIV;
    private Button buttonA, buttonB;
    private AnimationDrawable animationDrawable;

    //重写 onCreate 方法
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        requestWindowFeature(Window.FEATURE_NO_TITLE);
        setContentView(R.layout.main);
        animationIV = (ImageView) findViewById(R.id.animationIV);
        //获取按钮对象 buttonA,其 ID 为 buttonA
        buttonA = (Button) findViewById(R.id.buttonA);
        //获取按钮对象 buttonB,其 ID 为 buttonB
        buttonB = (Button) findViewById(R.id.buttonB);

        //为按钮对象 buttonA 设置监听器
        buttonA.setOnClickListener(new OnClickListener() {

            //重写 onClick 方法
```

```
35            @Override
36            public void onClick(View v) {
37                animationIV.setImageResource(R.anim.frame);
38                animationDrawable = (AnimationDrawable) animationIV.getDrawable();
39                animationDrawable.start();      //启动 AnimationDrawable
40            }
41        });
42
43        //为按钮对象 buttonB 设置监听器
44        buttonB.setOnClickListener(new OnClickListener() {
45
46            //重写 onClick 方法
47            @Override
48            public void onClick(View v) {
49                animationDrawable = (AnimationDrawable) animationIV.getDrawable();
50                animationDrawable.stop();       //停止 AnimationDrawable
51            }
52        });
53    }
54 }
```

① 第 14 行至第 54 行定义一个类 FrameAnimationExampleActivity 继承 Activity 类。

② 第 20 行至第 53 行重写 onCreate 方法。

③ 第 27 行获取按钮对象 buttonA，其 ID 为 buttonA，第 29 行获取按钮对象 buttonB，其 ID 为 buttonB。

④ 第 32 行至第 41 行为按钮对象 buttonA 设置监听器，第 35 行至第 40 行重写 onClick 方法，第 39 行通过 animationDrawable 对象的 start()方法启动逐帧动画。

⑤ 第 44 行至第 52 行为按钮对象 buttonB 设置监听器，第 47 行至第 51 行重写 onClick 方法，第 50 行通过 animationDrawable 对象的 stop()方法停止逐帧动画。

【运行结果】

在 Eclipse 中启动模拟器，然后运行项目 FrameAnimationExample，界面循环播放树叶叶片的逐帧动画，单击“逐帧动画停止”按钮停止播放，如图 8.4 所示，单击“逐帧动画显示”按钮继续播放，如图 8.5 所示。



图 8.4　单击“逐帧动画停止”按钮停止播放



图 8.5　单击“逐帧动画显示”按钮继续播放

8.3.2　补间动画

补间动画通过一系列的指令，将一个 View 对象进行位置、尺寸、旋转、透明度等变换从而形成动画，View 对象可以是图片、文本、按钮等对象。

补间动画可在 res/anim 目录中的 XML 的文件中进行定义，声明动画使用的变换、何时变换和持续时间等，补间动画常用的标记和属性如表 8.6 所示。

表 8.6　补间动画常用的标记和属性

标记名称	属性值		说明
< set >	shareInterpolator：在子元素中共享插入器		包含其他动画变换的容器
< alpha >	fromAlpha：起始透明度	toAlpha：终止透明度	取值为 0.0～1.0，其中 0.0 为全透明
< scale >	fromXScale：X 的起始值	toXScale：X 的终止值	实现尺寸变换，其中 1.0 为原始大小
	fromYScale：Y 的起始值	toYScale：Y 的终止值	
	pivotX：中心的 X 坐标	pivotY：中心的 Y 坐标	
< translate >	fromXDelta：X 起始位置	toXDelta：X 终止位置	实现水平或垂直移动，以"%"结尾代表相对于自身的比例；以"%p"结尾代表相对于父控件的比例；不以任何后缀结尾代表绝对值
	fromYDelta：Y 起始位置	toYDelta：Y 终止位置	
< rotate >	fromDegree：开始位置	toDegree：结束位置	实现旋转，可以指定旋转定位点
	pivotX：中心的 X 坐标	pivotY：中心的 Y 坐标	
< Interpolator >	无		插入器，描述变换的速度曲线

表 8.6 的标记中的一些公共属性如表 8.7 所示。

表 8.7　补间动画标记的常用公共属性

属　　性	说　　明
duration	变换持续的时间，以毫秒为单位
startOffset	变换开始的时间，以毫秒为单位
repeatCount	定义该动画重复的次数
interpolator	为每个子标记变换设置插入器，系统已经设置好一些插入器，可以在 R. anim 包下找到

【例 8.5】 制作旋转、透明度、缩放、平移补间动画。

【解题思路】

制作一个风景图片的旋转、透明度、缩放、平移补间动画，单击“旋转补间动画”按钮播放旋转补间动画，单击“透明度补间动画”按钮播放透明度补间动画，单击“缩放补间动画”按钮播放缩放补间动画，单击“平移补间动画”按钮播放平移补间动画。

【开发步骤和程序分析】

(1) 创建项目。在 Eclipse 中创建一个 TweenAnimationTest 应用项目，包名为 com. application. tweenanimationtest。

(2) 准备图片。准备好一个风景图片，命名为 scenery. png，复制到 res/drawable-mdpi 目录下。

(3) 定义动画。编写动画属性的描述代码，在 res 目录下新建 anim 目录，在该目录中创建定义旋转补间动画的 anim_rotate. xml 文件，创建定义透明度补间动画的 anim_alpha . xml 文件，创建定义缩放补间动画的 anim_scale. xml 文件，创建定义平移补间动画的 anim_translate. xml 文件。anim_rotate. xml 文件代码如下：

```
<?xml version = "1.0" encoding = "utf - 8"?>
< set xmlns:android = "http://schemas.android.com/apk/res/android" >

    <! -- 旋转变换,开始时旋转角度为 0 度,结束时旋转角度为 720 度 -->
    < rotate
        android:duration = "2000"
        android:fromDegrees = "0"
        android:interpolator = "@android:anim/accelerate_interpolator"
        android:pivotX = "50 % "
        android:pivotY = "50 % "
        android:toDegrees = "720" >
    </rotate >

    <! -- 旋转变换,开始时旋转角度为 360 度,结束时旋转角度为 0 度 -->
    < rotate
        android:duration = "2000"
        android:fromDegrees = "360"
        android:interpolator = "@android:anim/accelerate_interpolator"
        android:pivotX = "50 % "
```

```
            android:pivotY = "50 % "
            android:startOffset = "2000"
            android:toDegrees = "0" >
        </rotate>
</set>
```

anim_alpha. xml 文件代码如下：

```
<?xml version = "1.0" encoding = "utf - 8"?>
< set xmlns:android = "http://schemas.android.com/apk/res/android"
        android:fillAfter = "true"
        android:fillEnabled = "true" >

        <! -- 透明度的变换 -->
        < alpha
            android:duration = "4000"
            android:fromAlpha = "1"
            android:repeatCount = "1"
            android:repeatMode = "reverse"
            android:toAlpha = "0" />
</set>
```

anim_scale. xml 文件代码如下：

```
<?xml version = "1.0" encoding = "utf - 8"?>
< set  xmlns:android = "http://schemas.android.com/apk/res/android">

        <! -- 尺寸的变换 -->
        < scale
            android:fromXScale = "1"
            android:interpolator = "@android:anim/decelerate_interpolator"
            android:fromYScale = "1"
            android:toXScale = "2.0"
            android:toYScale = "2.0"
            android:pivotX = "50 % "
            android:pivotY = "50 % "
            android:fillAfter = "true"
            android:repeatCount = "1"
            android:repeatMode = "reverse"
            android:duration = "4000"/>
</set>
```

anim_translate. xml 文件代码如下：

```
<?xml version = "1.0" encoding = "utf - 8"?>
< set xmlns:android = "http://schemas.android.com/apk/res/android">

```

```
    <! -- 位置的变换 -->
    <translate
        android:fromXDelta = "0"
        android:toXDelta = "860"
        android:fromYDelta = "0"
        android:toYDelta = "0"
        android:fillAfter = "true"
        android:repeatMode = "reverse"
        android:repeatCount = "1"
        android:duration = "2000">
    </translate>
</set>
```

(4) 设计布局。在 res/layout 目录下的 main. xml 文件中,定义垂直线性布局 LinearLayout,在该布局中,设置一个内嵌的水平线性布局,其中分别设置“旋转补间动画”和“透明度补间动画”两个按钮;设置一个内嵌的水平线性布局,其中分别设置“缩放补间动画”和“平移补间动画”两个按钮;设置一个内嵌的线性布局,其中设置一个 ImageView 控件。在该文件中编辑代码如下:

```
<?xml version = "1.0" encoding = "utf - 8"?>

<! -- 声明一个垂直分布的线性布局 -->
<LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:layout_width = "match_parent"
    android:layout_height = "match_parent"
    android:orientation = "vertical" >

    <! -- 声明一个内嵌的水平分布线性布局 -->
    <LinearLayout
        android:orientation = "horizontal"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:layout_gravity = "center_horizontal"
    >

       <! -- 声明一个 Button 控件,其 ID 为 btnRotate,按钮名为"旋转补间动画" -->
       <Button
           android:id = "@ + id/btnRotate"
           android:layout_width = "fill_parent"
           android:layout_height = "wrap_content"
           android:text = "旋转补间动画" />

       <! -- 声明一个 Button 控件,其 ID 为 btnAlpha,按钮名为"透明度补间动画" -->
       <Button
           android:id = "@ + id/btnAlpha"
```

```
27              android:layout_width = "fill_parent"
28              android:layout_height = "wrap_content"
29              android:text = "透明度补间动画" />
30      </LinearLayout>
31
32      <! -- 声明一个内嵌的水平分布线性布局 -->
33      <LinearLayout
34          android:orientation = "horizontal"
35          android:layout_width = "wrap_content"
36          android:layout_height = "wrap_content"
37          android:layout_gravity = "center_horizontal"
38      >
39
40          <! -- 声明一个 Button 控件,其 ID 为 btnScale,按钮名为"缩放补间动画" -->
41          <Button
42              android:id = "@ + id/btnScale"
43              android:layout_width = "fill_parent"
44              android:layout_height = "wrap_content"
45              android:text = "缩放补间动画" />
46
47          <! -- 声明一个 Button 控件,其 ID 为 btnTranslate,按钮名为"平移补间动画" -->
48          <Button
49              android:id = "@ + id/btnTranslate"
50              android:layout_width = "fill_parent"
51              android:layout_height = "wrap_content"
52              android:text = "平移补间动画" />
53      </LinearLayout>
54
55      <LinearLayout
56          android:layout_width = "fill_parent"
57          android:layout_height = "fill_parent"
58          android:gravity = "center" >
59
60          <! -- 声明一个 ImageView 控件 -->
61          <ImageView
62              android:id = "@ + id/imageview"
63              android:layout_width = "150dp"
64              android:layout_height = "150dp"
65              android:src = "@drawable/scenery" />
66      </LinearLayout>
67  </LinearLayout>
```

第 65 行设置 ImageView 控件的图片源为 drawable 目录下的 scenery. png 文件。

(5) 编辑代码。在 com. application. tweenanimationtest 包下的 TweenAnimationTestActivity. java 文件中,定义一个类 TweenAnimationActivity 继承 Activity 类,重写 onCreate 方法,分

别获取按钮对象 button1、button2、button3、button4；为按钮对象 button1 设置监听器，通过 iv. startAnimation(alpha)方法启动透明度补间动画；为按钮对象 button2 设置监听器，通过 iv. startAnimation(rotate)方法启动旋转补间动画；为按钮对象 button3 设置监听器，通过 iv. startAnimation(scale)方法启动缩放补间动画；为按钮对象 button4 设置监听器，通过 iv. startAnimation(translate)方法启动平移补间动画。在该文件中编辑代码如下：

```
package com.application.tweenanimationexample;

import com.application.tweenanimationexample.R;
import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.ImageView;
import android.view.View.OnClickListener;
import android.view.animation.Animation;
import android.view.animation.AnimationUtils;

//定义一个类 TweenAnimationActivity 继承 Activity 类
public class TweenAnimationActivity extends Activity {
    private Button button1, button2, button3, button4;
    private boolean flag = true;

    //重写 onCreate()方法
    @Override
    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        final Animation rotate = AnimationUtils.loadAnimation(this, R.anim.anim_rotate);
        final Animation translate = AnimationUtils.loadAnimation(this, R.anim.anim_translate);
        final Animation scale = AnimationUtils.loadAnimation(this, R.anim.anim_scale);
        final Animation alpha = AnimationUtils.loadAnimation(this, R.anim.anim_alpha);
        final ImageView iv = (ImageView) findViewById(R.id.imageview);

        //获取按钮对象 button1,其 ID 为 btnAlpha
        button1 = (Button) findViewById(R.id.btnAlpha);
        //获取按钮对象 button2,其 ID 为 btnRotate
        button2 = (Button) findViewById(R.id.btnRotate);
        //获取按钮对象 button3,其 ID 为 btnScale
        button3 = (Button) findViewById(R.id.btnScale);
        //获取按钮对象 button4,其 ID 为 btnTranslate
        button4 = (Button) findViewById(R.id.btnTranslate);

        //为按钮对象 button1 设置监听器
```

```
        button1.setOnClickListener(new OnClickListener() {

            //重写 onClick 方法
            @Override
            public void onClick(View v) {
                if (flag) {
                    iv.startAnimation(alpha);
                }
            }
        });

        //为按钮对象 button2 设置监听器
        button2.setOnClickListener(new OnClickListener() {

            //重写 onClick ()方法
            @Override
            public void onClick(View v) {
                if (flag) {
                    iv.startAnimation(rotate);
                }
            }
        });

        //为按钮对象 button3 设置监听器
        button3.setOnClickListener(new OnClickListener() {

            //重写 onClick()方法
            @Override
            public void onClick(View v) {
                if (flag) {
                    iv.startAnimation(scale);
                }
            }
        });

        //为按钮对象 button4 设置监听器
        button4.setOnClickListener(new OnClickListener() {

            //重写 onClick()方法
            @Override
            public void onClick(View v) {
                if (flag) {
                    iv.startAnimation(translate);
                }
            }
```

```
85            });
86        }
87 }
```

① 第 14 行至第 87 行定义一个类 TweenAnimationActivity 继承 Activity 类。

② 第 19 行至第 86 行重写 onCreate 方法。

③ 第 31 行获取按钮对象 button1，其 ID 为 btnAlpha，第 33 行获取按钮对象 button2，其 ID 为 btnRotate，第 35 行获取按钮对象 button3，其 ID 为 btnScale，第 37 行获取按钮对象 button4，其 ID 为 btnTranslate。

④ 第 40 行至第 49 行为按钮对象 button1 设置监听器，第 43 行至第 48 行重写 onClick 方法，第 46 行通过 iv. startAnimation(alpha)方法启动透明度补间动画。

⑤ 第 52 行至第 61 行为按钮对象 button2 设置监听器，第 55 行至第 60 行重写 onClick 方法，第 58 行通过 iv. startAnimation(rotate)方法启动旋转补间动画。

⑥ 第 64 行至第 73 行为按钮对象 button3 设置监听器，第 67 行至第 72 行重写 onClick 方法，第 70 行通过 iv. startAnimation(scale)方法启动缩放补间动画。

⑦ 第 76 行至第 85 行为按钮对象 button4 设置监听器，第 79 行至第 84 行重写 onClick 方法，第 82 行通过 iv. startAnimation(translate)方法启动平移补间动画。

【运行结果】

在 Eclipse 中启动模拟器，然后运行项目 TweenAnimationTest，初始画面如图 8.6 所示。单击“旋转补间动画”按钮后，播放旋转补间动画如图 8.7 所示。

单击“透明度补间动画”按钮后，播放透明度补间动画如图 8.8 所示。单击“缩放补间动画”按钮后，播放缩放补间动画如图 8.9 所示。



图 8.6　初始画面



图 8.7　旋转补间动画



图 8.8　透明度补间动画



图 8.9　缩放补间动画

单击“平移补间动画”按钮后，播放平移补间动画如图 8.10 所示。



图 8.10　平移补间动画

【例 8.6】 制作补间动画举例。

【解题思路】

制作一个鹰的补间动画，单击“补间动画”按钮后，播放一个从小到大、从暗到亮、绕中心旋转的鹰的动画。

【开发步骤和程序分析】

(1) 创建项目。在 Eclipse 中创建一个 TweenAnimationExample 应用项目，包名为 com.application.tweenanimationexample。

(2) 准备图片。准备好一个鹰的图片，命名为 eagle.png，复制到 res/drawable-mdpi 目录下。

(3) 定义动画。在 res 目录下新建 anim 目录，在该目录中创建 tween_animation.xml 文件，tween_animation.xml 文件代码如下：

```
<?xml version = "1.0" encoding = "utf - 8"?>
< set xmlns:android = "http://schemas.android.com/apk/res/android">

<! -- 透明度的变换 -->
< alpha
    android:fromAlpha = "0.0"
    android:toAlpha = "1.0"
    android:duration = "5000"
    />
  <! -- 尺寸的变换 -->
  < scale
        android:interpolator = "@android:anim/accelerate_decelerate_interpolator"
        android:fromXScale = "0.0"
        android:toXScale = "1.0"
        android:fromYScale = "0.0"
        android:toYScale = "1.0"
        android:pivotX = "50 % "
        android:pivotY = "50 % "
        android:fillAfter = "false"
        android:duration = "9000"
    />
    <! -- 位置的变换 -->
    < translate
        android:fromXDelta = "30"
        android:toXDelta = "0"
        android:fromYDelta = "30"
        android:toYDelta = "0"
        android:duration = "10000"
    />
    <! -- 旋转变换 -->
    < rotate
        android:interpolator = "@android:anim/accelerate_decelerate_interpolator"
        android:fromDegrees = "0"
```

```
            android:toDegrees = " + 360"
            android:pivotX = "50 % "
            android:pivotY = "50 % "
            android:duration = "10000"
        />
</set >
```

(4) 设计布局。在 res/layout 目录下的 main. xml 文件中，声明一个垂直分布的线性布局，在该布局中，设置一个 ImageView 控件，设置一个“补间动画”按钮。在该文件中编辑代码如下：

```
1  <?xml version = "1.0" encoding = "utf - 8"?>
2
3  <! -- 声明一个垂直分布的线性布局 -- >
4  < LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
5      android:orientation = "vertical"
6      android:layout_width = "fill_parent"
7      android:layout_height = "fill_parent"
8  >
9
10     <! -- 声明一个 ImageView 控件 -- >
11     < ImageView
12         android:id = "@ + id/iv"
13         android:src = "@drawable/eagle"
14         android:layout_width = "wrap_content"
15         android:layout_height = "wrap_content"
16         android:layout_gravity = "center_horizontal"
17         />
18
19     <! -- 声明一个 Button 控件 -- >
20     < Button
21         android:id = "@ + id/btn1"
22         android:text = "补间动画"
23         android:layout_width = "100px"
24         android:layout_height = "wrap_content"
25         android:layout_gravity = "center_horizontal"
26         android:layout_marginRight = "10dip"
27         />
28 </LinearLayout >
```

第 13 行设置 ImageView 控件的图片源为 drawable 目录下的 eagle. png 文件。

(5) 编辑代码。在 com. application. tweenanimationexample 包下的 TweenAnimationExampleActivity. java 文件中，定义一个类 TweenAnimationExampleActivity 继承 Activity 类，重写 onCreate 方法，获取按钮对象 btn1，为按钮对象 btn1 设置监听器，重写 onClick 方法，通过 iv. startAnimation(animation) 方法启动补间动画。在该文件中编辑代码如下：

```java
1  package com.application.tweenanimationexample;
2
3  import com.application.tweenanimationexample.R;
4  import android.app.Activity;
5  import android.os.Bundle;
6  import android.view.View;
7  import android.view.View.OnClickListener;
8  import android.view.animation.Animation;
9  import android.view.animation.AnimationUtils;
10 import android.widget.Button;
11 import android.widget.ImageView;
12
13 //定义一个类 TweenAnimationExampleActivity 继承 Activity 类
14 public class TweenAnimationExampleActivity extends Activity {
15
16     //重写 onCreate()方法
17     @Override
18     public void onCreate(Bundle savedInstanceState) {
19         super.onCreate(savedInstanceState);
20         setContentView(R.layout.main);                    //设置屏幕
21         Button btn1 = (Button)findViewById(R.id.btn1);    //获取 Button 对象
22
23         //为 Button 对象添加 OnClickListener 监听器
24         btn1.setOnClickListener(new OnClickListener() {
25
26             //重写 onClick()方法
27              @Override
28             public void onClick(View v) {
29                 ImageView iv = (ImageView)findViewById(R.id.iv);
30                 iv.setImageDrawable(getResources().getDrawable(R.drawable.eagle));
31                 Animation animation = AnimationUtils.loadAnimation(
32                         TweenAnimationExampleActivity.this, R.anim. tween_animation);
33                 iv.startAnimation(animation);             //启动补间动画
34              }
35         });
36     }
37 }
```

① 第 14 行至第 37 行定义一个类 TweenAnimationExampleActivity 继承 Activity 类。

② 第 17 行至第 36 行重写 onCreate 方法。

③ 第 21 行获取按钮对象 btn1，其 ID 为 btn1。

④ 第 24 行至第 35 行为按钮对象 btn1 设置监听器，第 27 行至第 34 行重写 onClick 方法，第 33 行通过 iv.startAnimation(animation)方法启动补间动画。

【运行结果】

在 Eclipse 中启动模拟器，然后运行项目 TweenAnimationExampleActivity，初始画面

如图 8.11 所示。单击“补间动画”按钮后，播放一个从小到大、从暗到亮、绕中心旋转的鹰的动画，如图 8.12 和图 8.13 所示。



图 8.11　初始画面

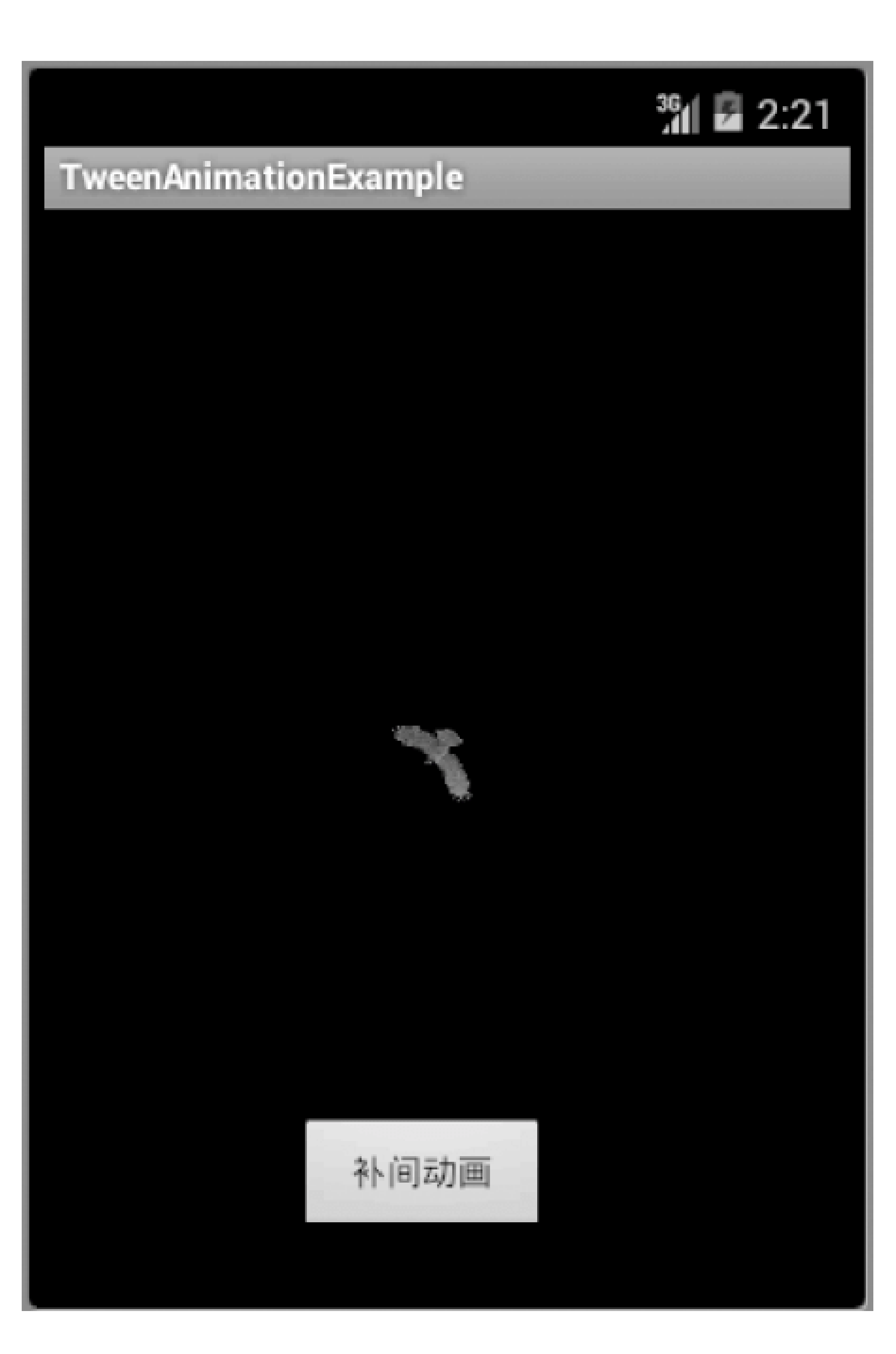


图 8.12　补间动画画面 1



图 8.13　补间动画画面 2

8.4 音频播放与视频播放

音频播放常用 MediaPlayer 类和使用 SoundPool 类，视频播放常用 VideoView，MediaPlayer 和 SurfaceView，下面分别介绍。

8.4.1 音频播放

Android 播放音频的方式有两种：使用 MediaPlayer 类和使用 SoundPool 类。使用 MediaPlayer 类用于播放短促、反应速度快的声音，例如游戏中的音效配音，而 SoundPool 常用于播放较长的、对反应时间要求不高的声音，例如播放后台音乐、歌曲等。

Android 的音频文件存放在项目的 res/raw 文件夹下，Android 支持的音频格式有 MP3、MID、WAV、OGG、AMR 等，音频格式采样率为 11/22/44.1kHz，16 位立体声。

1. 使用 MediaPlayer 类播放音频

使用 MediaPlayer 类可用于播放大容量的音频文件，支持多种播放操作，支持与媒体操作相关的监听器。

1）创建 MediaPlayer 对象

创建 MediaPlayer 对象有两种方式：使用 new MediaPlayer()和使用 MediaPlayer.create()。

2）使用 MediaPlayer 监听器

MediaPlayer 对象可以设置的监听器有 OnCompletionListener、OnPrepareListener、OnErrorListener、OnBufferingUpdateListener、OnInfoListener，OnVideoSizeChangedListener 和 OnSeekCompleteListener 等。

2. 使用 SoundPool 类播放音频

使用 SoundPool 类播放音频的步骤：

1）创建一个 SoundPool 对象

创建 SoundPool 对象的方法为：

```
SoundPool(int maxStream, int streamType, int srcQuality)
```

2）加载音频资源

SoundPool 通过 load()方法来加载音频资源，load()方法有 4 种方式加载音频：通过一个 AssetFileDescriptor 对象加载音频，通过一个资源 ID 加载音频，通过指定的路径加载音频，通过 FileDescriptor 加载音频。

3）播放控制

play()方法传递的是一个 load()返回的 soundID，它指向一个被记载的音频资源，而 pause()、resume()和 stop()方法是针对播放流操作的。

【例 8.7】 音频播放举例。

【解题思路】

使用 MediaPlayer 和 SoundPool 播放音频，两种音频可以单独播放，也可以同时播放，

音频文件放在项目的 res/raw 文件夹下。

【开发步骤和程序分析】

(1) 在 Eclipse 中创建一个 AudioPlayExample 应用项目,包名为 com. application. audioplayexample。

(2) 设计布局。在 res/layout 目录下的 main. xml 文件中,定义垂直线性布局 LinearLayout,在该布局中,设置 MediaPlayer 文本框,设置一个内嵌的水平线性布局,其中分别设置"播放声音"和"暂停播放"两个按钮;设置"SoundPool"文本框,设置一个内嵌的水平线性布局,其中分别设置"播放声音"和"暂停播放"两个按钮。在该文件中编辑代码如下:

```
<?xml version = "1.0" encoding = "utf - 8"?><?xml version = "1.0" encoding = "utf - 8"?>
<! -- 定义垂直分布的线性布局 -->
<LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = "vertical"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    >
    <TextView
        android:id = "@ + id/textView"
        android:layout_width = "fill_parent"
        android:layout_height = "wrap_content"
        android:text = "未播放任何声音"
       />

       <! -- 设置"MediaPlayer"文本框 -->
       <TextView
           android:id = "@ + id/mp"
           android:layout_width = "fill_parent"
           android:layout_height = "wrap_content"
           android:text = "MediaPlayer"
          />
       <LinearLayout android:id = "@ + id/LinearLayout01"
           android:orientation = "horizontal"
           android:layout_width = "wrap_content"
           android:layout_height = "wrap_content"
           android:layout_gravity = "center_horizontal"
           >

           <! -- 设置"播放声音"按钮 -->
           <Button
                android:id = "@ + id/button1"
                android:layout_width = "fill_parent"
                android:layout_height = "wrap_content"
                android:text = "播放声音"
```

```
                    />
                <!-- 设置"暂停播放"按钮 -->
                <Button
                        android:id="@+id/button2"
                        android:layout_width="fill_parent"
                        android:layout_height="wrap_content"
                        android:text="暂停播放"
                    />
        </LinearLayout>

        <!-- 设置"SoundPool"文本框 -->
        <TextView
            android:id="@+id/sp"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="SoundPool"
            />

    <LinearLayout android:id="@+id/LinearLayout02"
            android:orientation="horizontal"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center_horizontal"
            >

            <!-- 设置"播放声音"按钮 -->
            <Button
                    android:id="@+id/button3"
                    android:layout_width="fill_parent"
                    android:layout_height="wrap_content"
                    android:text="播放声音"
                />
            <!-- 设置"暂停播放"按钮 -->
                <Button
                    android:id="@+id/button4"
                    android:layout_width="fill_parent"
                    android:layout_height="wrap_content"
                    android:text="暂停播放"
                />
        </LinearLayout>
</LinearLayout>
```

(3) 在 com.application.audioplayexample 包下的 AudioPlayExampleActivity.java 文件中，定义一个类 AudioPlayExampleActivity 继承 Activity 类，且实现事件监听器接口，定义 initSounds()方法，初始化播放声音对象，初始化 MediaPlayer 对象，初始化 SoundPool 对

象,并创建 HashMap 对象存放多个音频文件;实现 onClick()方法,单击有关按钮,分别执行 MediaPlayer 和 SoundPool 有关播放和暂停控制操作。在该文件中编辑代码如下:

```
package com.application.audioplayexample;

import java.util.HashMap;
import com.application.audioplayexample.R;
import android.app.Activity;
import android.content.Context;
import android.media.AudioManager;
import android.media.MediaPlayer;
import android.media.SoundPool;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.TextView;

   //定义一个类 AudioPlayExampleActivity 继承 Activity 类,且实现事件监听器接口
   public class AudioPlayExampleActivity extends Activity implements OnClickListener{
       Button button1;
       Button button2;
       Button button3;
       Button button4;
       TextView textView;
       MediaPlayer mMediaPlayer;
       HashMap<Integer, Integer> soundPoolMap;

       //重写 onCreate 方法
       @Override
       public void onCreate(Bundle savedInstanceState) {
           super.onCreate(savedInstanceState);
           initSounds();                              //调用 initSounds()方法

           //调用 setContentView()方法引用 main 布局
           setContentView(R.layout.main);

           //获取控件的唯一标识
           textView = (TextView) this.findViewById(R.id.textView);
           button1 = (Button) this.findViewById(R.id.button1);
           button2 = (Button) this.findViewById(R.id.button2);
           button3 = (Button) this.findViewById(R.id.button3);
           button4 = (Button) this.findViewById(R.id.button4);

           //为按钮对象设置监听器
```

```
        button1.setOnClickListener(this);
        button2.setOnClickListener(this);
        button3.setOnClickListener(this);
        button4.setOnClickListener(this);
    }

    //定义 initSounds()方法
    public void initSounds(){
        //初始化 MediaPlayer 对象
        mMediaPlayer = MediaPlayer.create(this, R.raw.starbackgrand);
        //初始化 SoundPool 对象
        soundPool = new SoundPool(4, AudioManager.STREAM_MUSIC, 100);
        soundPoolMap = new HashMap<Integer, Integer>();
        soundPoolMap.put(1, soundPool.load(this, R.raw.backsound, 1));
    }

    //实现 onClick()方法
    public void onClick(View v) {
        if(v == button1){              //在 MediaPlayer 下方,单击"播放声音"按钮
            textView.setText("使用 MediaPlayer 播放声音");
            if(!mMediaPlayer.isPlaying()){
                mMediaPlayer.start();  //播放声音
            }
        }
        else if(v == button2){         //在 MediaPlayer 下方,单击"暂停播放"按钮
            textView.setText("暂停声音播放");
            if(mMediaPlayer.isPlaying()){
                mMediaPlayer.pause();  //暂停播放
            }
        }
        else if(v == button3){         //在 SoundPool 下方,单击"播放声音"按钮
            textView.setText("使用 SoundPool 播放声音");
            this.playSound(1, 0);      //播放声音
        }
        else if(v == button4){         //在 SoundPool 下方,单击"暂停播放"按钮
            textView.setText("暂停声音播放");
            soundPool.pause(1);        //暂停播放
        }
    }
    //用 SoundPoll 播放声音的方法
    public void playSound(int sound, int loop) {
        AudioManager mgr = (AudioManager)this.getSystemService(Context.AUDIO_SERVICE);
        float streamVolumeCurrent = mgr.getStreamVolume(AudioManager.STREAM_MUSIC);
        float streamVolumeMax = mgr.getStreamMaxVolume(AudioManager.STREAM_MUSIC);
        float volume = streamVolumeCurrent/streamVolumeMax;
```

```
88                //播放声音
89                soundPool.play(soundPoolMap.get(sound), volume, volume, 1, loop, 1f);
90          }
91    }
```

① 第 16 行至第 91 行定义一个类 AudioPlayExampleActivity 继承 Activity 类，且实现事件监听器接口。

② 第 50 行至第 57 行实现定义 initSounds()方法，初始化播放声音对象，第 52 行初始化 MediaPlayer 对象，第 54 行至第 56 行初始化 SoundPool 对象，并创建 HashMap 对象存放多个音频文件。

③ 第 60 行至第 81 行实现 onClick()方法，单击有关按钮，分别执行 MediaPlayer 和 SoundPool 有关播放和暂停控制操作。

【运行结果】

在 Eclipse 中启动模拟器，然后运行项目 AudioPlayExample，界面如图 8.14 所示，在 MediaPlayer 栏和 SoundPool 栏，单击有关按钮则可进行声音播放和暂停。

图 8.14　音频播放界面

8.4.2　视频播放

可以使用 MediaPlayer 和 SurfaceView 播放视频，也可以使用 VideoView 播放视频。

Android 支持的视频格式有 MP4、3GP、H.263、H.264(AVC)等。

比较大的视频文件常存放在 SD 卡中，在模拟器中使用 SD 卡的步骤如下：

(1) 创建一个 SD 卡镜像文件。

(2) 关联 SD 卡和模拟器。

(3) 向 SD 卡中导入文件。

(4) 在模拟器中使用 SD 卡中的文件。

1. 使用 MediaPlayer 和 SurfaceView 播放视频

需要为 MediaPlayer 播放器创建一个用于绘制图像的控件 Surface，采用 MediaPlayer 与 SurfaceView 结合用来实现视频的输出。

使用 MediaPlayer 和 SurfaceView 的步骤如下：

(1) 创建 MediaPlayer 对象，并设置加载的视频文件(使用 setDataSource()方法)。

(2) 在界面布局文件中定义 SurfaceView 控件。

(3) 通过 MediaPlayer.setDisplay(SurfaceHolder sh)来指定视频画面输出到 SurfaceView 之上。

(4) 通过 MediaPlayer 的方法播放视频。

2. 使用 VideoView 播放视频

在 XML 文件中加入 VideoView 控件，再从 SD Card 中载入 MP4 文件或 3GP 文件，就能使用 VideoView 播放视频。

【例 8.8】 视频播放举例。

【解题思路】

使用 MediaPlayer 和 SurfaceView 播放视频，通过播放视频按钮和暂停播放按钮控制视频播放。

【开发步骤和程序分析】

(1) 在 Eclipse 中创建一个 VideoPlayExample 应用项目，包名为 com. application. videoplayexample。

(2) 设计布局。在布局中，设置一个 SurfaceView 控件，用于播放视频。

在 res/layout 目录下的 main. xml 文件中，定义垂直线性布局 LinearLayout，在该布局中，设置 SurfaceView，用于播放视频，设置一个内嵌的线性布局，其中分别设置“播放视频”和“暂停播放”两个按钮。在该文件中编辑代码如下：

```
<?xml version = "1.0" encoding = "utf - 8"?>

<! -- 定义垂直分布的线性布局 -->
< LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = "vertical"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent"
    >

        <! -- 设置 SurfaceView, 用于播放视频 -->
        < SurfaceView
                android:id = "@ + id/surfaceView"
                android:layout_width = "match_parent"
                android:layout_height = "320px"
               />
        < LinearLayout
                android:layout_width = "fill_parent"
                android:layout_height = "wrap_content"
                android:gravity = "center"
               >

            <! -- 设置"播放视频"按钮 -->
            < Button
                android:id = "@ + id/play2_Button"
                android:layout_width = "wrap_content"
                android:layout_height = "wrap_content"
                android:text = "播放视频"
               />
```

```

            <!-- 设置"暂停播放"按钮 -->
            <Button
                android:id="@+id/pause2_Button"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="暂停播放"
               />
        </LinearLayout>
</LinearLayout>
```

（3）在 com.application.videoplayexample 包下的 VideoPlayExampleActivity 文件中，定义一个类 VideoPlayExampleActivity 继承 Activity 类，且实现两个接口：OnClickListener，SurfaceHolder.Callback，设置视频文件的路径，重写 onCreate 方法，定义 playVideo()方法，重写 SurfaceHolder.Callback 接口的三个抽象方法。在该文件中编辑代码如下：

```
package com.application.videoplayexample;

import java.io.File;
import java.io.IOException;
import com.application.videoplayexample.R;
import android.app.Activity;
import android.graphics.PixelFormat;
import android.media.AudioManager;
import android.media.MediaPlayer;
import android.os.Bundle;
import android.os.Environment;
import android.view.SurfaceHolder;
import android.view.SurfaceView;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.Toast;
    //定义一个类 VideoPlayExampleActivity 继承 Activity 类,且实现两个接口:
    //OnClickListener, SurfaceHolder.Callback
    public class VideoPlayExampleActivity extends Activity implements
            OnClickListener, SurfaceHolder.Callback {
        //设置视频文件的路径
        String path = "/sdcard/video.mp4";
        Button play_Button;
        Button pause_Button;
        boolean isPause = false;
        SurfaceView surfaceView;
        SurfaceHolder surfaceHolder;
```

```java
        MediaPlayer mediaPlayer;

        //重写 onCreate 方法
        public void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.main);
            play_Button = (Button) findViewById(R.id.play2_Button);
            play_Button.setOnClickListener(this);
            pause_Button = (Button) findViewById(R.id.pause2_Button);
            pause_Button.setOnClickListener(this);
            getWindow().setFormat(PixelFormat.UNKNOWN);
            surfaceView = (SurfaceView) findViewById(R.id.surfaceView);
            surfaceHolder = surfaceView.getHolder();
            surfaceHolder.addCallback(this);
            surfaceHolder.setFixedSize(176, 144);
            surfaceHolder.setType(SurfaceHolder.SURFACE_TYPE_PUSH_BUFFERS);
            mediaPlayer = new MediaPlayer();

            Toast.makeText(
                        this,
                        fileIsExists(Environment.getExternalStorageDirectory()
                            + "/video.mp4") + "", Toast.LENGTH_LONG).show();
        }

        public boolean fileIsExists(String strFile) {
            try {
                File f = new File(strFile);
                if (!f.exists()) {
                    return false;
                }

            } catch (Exception e) {
                return false;
            }

            return true;
        }

        public void onClick(View v) {
            if (v == play_Button) {        //单击"播放视频"按钮
                isPause = false;
                try {
                    playVideo(path);
                } catch (IOException e) {
                    //TODO Auto - generated catch block
```

```
74                      e.printStackTrace();
75                  }
76              } else if (v == pause_Button) {//单击"暂停播放"按钮,
77                  if (isPause == false) {    //如果正在播放则将其暂停
78                      mediaPlayer.pause();
79                      isPause = true;
80                  } else {                   //如果暂停则继续播放
81                      mediaPlayer.start();
82                      isPause = false;
83                  }
84              }
85          }
86
87          private void playVideo(String strPath) throws IOException {
                                                                    //自定义播放影片函数
88              mediaPlayer.reset();
89              mediaPlayer.setAudioStreamType(AudioManager.STREAM_MUSIC);
90              mediaPlayer.setDataSource(strPath);
91              mediaPlayer.setDisplay(surfaceHolder);
                                       //设置 Video 影片以 SurfaceHolder 播放
92              mediaPlayer.prepare();
93              mediaPlayer.start();
94
95          }
96
97          public void surfaceChanged(SurfaceHolder arg0, int arg1, int arg2, int arg3) {
98          }
99
100       public void surfaceCreated(SurfaceHolder arg0) {
101       }
102
103       public void surfaceDestroyed(SurfaceHolder arg0) {
104       }
105   }
```

① 第 20 行至第 105 行定义一个类 VideoPlayExampleActivity 继承 Activity 类，且实现两个接口：OnClickListener，SurfaceHolder.Callback。

② 第 23 行设置视频文件的路径。

③ 第 32 行至第 51 行重写 onCreate 方法，其中，第 39 行 getWindow().setFormat(PixelFormat.UNKNOWN)方法允许对窗口显示格式进行处理，第 42 行为 surfaceHolder 对象添加回调方法，第 43 行通过 surface 的 holder 设置显示尺寸，第 44 行设置显示类型。

④ 第 87 行至第 95 行定义 playVideo()方法。

⑤ 第 97 行至第 104 行重写 SurfaceHolder.Callback 接口的三个抽象方法。

【运行结果】

在 Eclipse 中启动模拟器，然后运行项目 VideoPlayExample，视频播放初始界面如图 8.15 所示，单击有关按钮则可进行视频播放和暂停。



图 8.15　视频播放初始界面

8.5　声音采集与图像采集

8.5.1　声音采集

声音采集通过 MediaRecorder 类来实现，MediaRecorder 包含了 Audio 和 video 的记录功能，录制语音文件通常用.amr 格式，在 AndroidManifest.xml 中要添加录音等权限。

常用的录音接口如下：

- 设置麦克风

recorder.setAudioSource(MediaRecorder.AudioSource.MIC);

- 设置输出格式

recorder.setOutputFormat(MediaRecorder.OutputFormat.THREE_GPP);

- 设置音频编码

Encoder recorder.setAudioEncoder(MediaRecorder.AudioEncoder.AMR_NB);

- 设置音频文件保存路径

recorder.setOutputFile("/sdcard/ataaw/ataaw.3gp");

- 开始录制

recorder. start();

【例 8.9】 声音采集举例。

【解题思路】

使用 MediaRecorder 类实现一个录音器，具有录音、停止、播放、删除等功能，录制的文件存放在 SD 卡中。

【开发步骤和程序分析】

(1) 在 Eclipse 中创建一个 SoundRecordExample 应用项目，包名为 com. application. soundrecordexample。

(2) 设计布局，在屏幕上部设置两行图片按钮，第一行为“录音”图片按钮、“停止”图片按钮，第二行为“播放”图片按钮、“删除”图片按钮，在屏幕中部设置一行文本用于显示当前音频文件，在屏幕下部设置一个列表视图列出已录制的音频文件。

在 res/layout 目录下的 main. xml 文件中，编辑代码如下：

```
<?xml version = "1.0" encoding = "utf - 8"?>

<! -- 定义垂直分布的线性布局 -->
< LinearLayout
      xmlns:android = "http://schemas.android.com/apk/res/android"
      android:orientation = "vertical"
      android:layout_width = "fill_parent"
      android:layout_height = "fill_parent"
      android:background = "@color/lightpurple">
      < LinearLayout
          android:id = "@ + id/LinearLayout01"
          android:layout_width = "wrap_content"
          android:layout_height = "wrap_content">

          <! -- 设置"录音"图片按钮 -->
          < ImageButton
               android:id = "@ + id/ImageButton01"
               android:layout_width = "wrap_content"
               android:layout_height = "wrap_content"
               android:src = "@drawable/recordsr">
          </ImageButton >

        <! -- 设置"停止"图片按钮 -->
        < ImageButton
               android:id = "@ + id/ImageButton02"
               android:layout_width = "wrap_content"
               android:layout_height = "wrap_content"
               android:src = "@drawable/stopsr">
        </ImageButton >
      </LinearLayout >
```

```
        <LinearLayout
            android:id = "@ + id/LinearLayout02"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content">

          <! -- 设置"播放"图片按钮 -->
          <ImageButton
                android:id = "@ + id/ImageButton03"
                android:layout_width = "wrap_content"
                android:layout_height = "wrap_content"
                android:src = "@drawable/playsr">
          </ImageButton>

          <! -- 设置"删除"图片按钮 -->
          <ImageButton
                android:id = "@ + id/ImageButton04"
                android:layout_width = "wrap_content"
                android:layout_height = "wrap_content"
                android:src = "@drawable/deletesr">
          </ImageButton>
        </LinearLayout>
        <TextView
            android:id = "@ + id/TextView01"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:textColor = "@color/black">
        </TextView>

        <! -- 声明 ListView 控件 -->
        <ListView
            android:id = "@ + id/ListView01"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:background = "@color/black">
        </ListView>
    </LinearLayout>
```

(3) 设计 ListView 的条目布局，在 res/layout 目录下的 my_list_item.xml 文件中，编辑代码如下：

```
<?xml version = "1.0" encoding = "utf - 8"?>
<CheckedTextView
    xmlns:android = "http://schemas.android.com/apk/res/android"
    android:id = "@ + id/myCheckedTextView1"
    android:layout_width = "fill_parent"
```

```
    android:layout_height = "fill_parent"
    android:textColor = "@color/white"/>
```

（4）在 com.application.soundrecordexample 包下的 SoundRecordExampleActivity.java 文件中，定义一个类 SoundRecordExampleActivity 继承 Activity 类，创建 ArrayAdapter 类的对象 adapter，将 ArrayAdapter 添加到 ListView 对象中；创建录制的音频文件，设置录音的来源、格式、音频编码、音频文件保存路径，设置完成后开始录音，设置图片按钮的可用状态；停止录音；播放录的制音频文件；删除录制的音频文件。在该文件中编辑代码如下：

```
package com.application.soundrecordexample;

import java.io.File;
import java.io.IOException;
import java.util.ArrayList;
import com.application.soundrecordexample.R;
import android.app.Activity;
import android.content.Intent;
import android.media.MediaRecorder;
import android.net.Uri;
import android.os.Bundle;
import android.os.Environment;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.CheckedTextView;
import android.widget.ImageButton;
import android.widget.ListView;
import android.widget.TextView;
import android.widget.Toast;

//定义一个类 SoundRecordExampleActivity 继承 Activity 类
public class SoundRecordExampleActivity extends Activity{
    private ImageButton myButton1;
    private ImageButton myButton2;
    private ImageButton myButton3;
    private ImageButton myButton4;
    private ListView myListView1;
    private String strTempFile = "voice_";
    private File myRecAudioFile;
    private File myRecAudioDir;
    private File myPlayFile;
    private MediaRecorder mMediaRecorder;
    private ArrayList<String> recordFiles;
    private ArrayAdapter<String> adapter;
```

```java
    private TextView myTextView1;
    private boolean sdCardExit;
    private boolean isStopRecord;

    //重写 onCreate 方法
    @Override
    public void onCreate(Bundle savedInstanceState){
      super.onCreate(savedInstanceState);

      //调用 setContentView()方法引用 main 布局
      setContentView(R.layout.main);

      //获取控件的唯一标识
      myButton1 = (ImageButton) findViewById(R.id.ImageButton01);
      myButton2 = (ImageButton) findViewById(R.id.ImageButton02);
      myButton3 = (ImageButton) findViewById(R.id.ImageButton03);
      myButton4 = (ImageButton) findViewById(R.id.ImageButton04);
      myListView1 = (ListView) findViewById(R.id.ListView01);
      myTextView1 = (TextView) findViewById(R.id.TextView01);
      myButton2.setEnabled(false);
      myButton3.setEnabled(false);
      myButton4.setEnabled(false);

      /* 判断 SD Card 是否插入 */
      sdCardExit = Environment.getExternalStorageState().equals(
          android.os.Environment.MEDIA_MOUNTED);
      /* 取得 SD Card 路径作为录音的文件位置 */
      if (sdCardExit)
        myRecAudioDir = Environment.getExternalStorageDirectory();

      /* 取得 SD Card 目录里的所有.amr 文件 */
      getRecordFiles();

      adapter = new ArrayAdapter<String>(this,
          R.layout.my_list_item, recordFiles);
      /* 将 ArrayAdapter 添加 ListView 对象中 */
      myListView1.setAdapter(adapter);

      /* 录音 */
      myButton1.setOnClickListener(new ImageButton.OnClickListener(){
        @Override
        public void onClick(View arg0){
          try{
            if (!sdCardExit){
              Toast.makeText(SoundRecordExampleActivity.this, "请插入 SD Card",
```

```
                    Toast.LENGTH_LONG).show();
                  return;
                }

                /* 创建录制的音频文件 */
                myRecAudioFile = File.createTempFile(strTempFile, ".amr", myRecAudioDir);
                mMediaRecorder = new MediaRecorder();
                /* 设置录音来源为麦克风 */
                mMediaRecorder.setAudioSource(MediaRecorder.AudioSource.MIC);
                mMediaRecorder.setOutputFormat(MediaRecorder.OutputFormat.DEFAULT);
                mMediaRecorder.setAudioEncoder(MediaRecorder.AudioEncoder.DEFAULT);
                mMediaRecorder.setOutputFile(myRecAudioFile.getAbsolutePath());
                mMediaRecorder.prepare();
                mMediaRecorder.start();
                myTextView1.setText("录音中");
                myButton2.setEnabled(true);
                myButton3.setEnabled(false);
                myButton4.setEnabled(false);
                isStopRecord = false;
              }
              catch (IOException e){
                e.printStackTrace();
              }
            }
          });

          /* 停止 */
          myButton2.setOnClickListener(new ImageButton.OnClickListener(){
            @Override
            public void onClick(View arg0){
              if (myRecAudioFile != null){
                /* 停止录音 */
                mMediaRecorder.stop();
                mMediaRecorder.release();
                mMediaRecorder = null;
                /* 将录制的音频文件名给 Adapter */
                adapter.add(myRecAudioFile.getName());
                myTextView1.setText("停止：" + myRecAudioFile.getName());
                myButton2.setEnabled(false);
                isStopRecord = true;
              }
            }
          });

          /* 播放 */
```

```
        myButton3.setOnClickListener(new ImageButton.OnClickListener(){
          @Override
          public void onClick(View arg0){
            if (myPlayFile != null && myPlayFile.exists()){
              /* 打开播放的程序 */
              openFile(myPlayFile);
            }
          }
        });

        /* 删除 */
        myButton4.setOnClickListener(new ImageButton.OnClickListener(){
          @Override
          public void onClick(View arg0){
            if (myPlayFile != null){
              /* 先将 Adapter 删除文件名 */
              adapter.remove(myPlayFile.getName());
              /* 删除文件 */
              if (myPlayFile.exists())
                myPlayFile.delete();
              myTextView1.setText("完成删除");
              /* 判断 adapter 为空时按钮的状态 */
              if (adapter.isEmpty()){
                myButton3.setEnabled(false);
                myButton4.setEnabled(false);
              };
            }
          }
        });

      /* 定义 ListView 的条目被单击的监听器  */
        myListView1.setOnItemClickListener(
          new AdapterView.OnItemClickListener(){
            @Override
            public void onItemClick(AdapterView<?> arg0, View arg1,
                int arg2, long arg3){
              /* 当有单击列表中的文件名时将"删除"及"播放"按钮置为 Enable */
              myButton3.setEnabled(true);
              myButton4.setEnabled(true);
              myPlayFile = new File(myRecAudioDir.getAbsolutePath()
                             + File.separator
                             + ((CheckedTextView) arg1).getText());
              myTextView1.setText("你选的是："
                             + ((CheckedTextView) arg1).getText());
            }
```

```
171             });
172         }
173
174         @Override
175         protected void onStop(){
176             if (mMediaRecorder != null && !isStopRecord){
177                 /* 停止录音 */
178                 mMediaRecorder.stop();
179                 mMediaRecorder.release();
180                 mMediaRecorder = null;
181             }
182             super.onStop();
183         }
184
185         /* 向 ArrayList 对象中添加 SD 卡中的.amr 文件名 */
186         private void getRecordFiles(){
187             recordFiles = new ArrayList<String>();
188             if (sdCardExit){
189                 File files[] = myRecAudioDir.listFiles();
190                 if (files != null){
191                     for (int i = 0; i < files.length; i++){
192                         if (files[i].getName().indexOf(".") >= 0){
193                             /* 只取.amr 文件 */
194                             String fileS = files[i].getName().substring(files[i].getName().indexOf("."));
195                             if (fileS.toLowerCase().equals(".amr"))
196                                 recordFiles.add(files[i].getName());
197                         }
198                     }
199                 }
200             }
201         }
202
203         /* 打开播放录音文件的程序 */
204         private void openFile(File f) {
205             Intent intent = new Intent();
206             intent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
207             intent.setAction(android.content.Intent.ACTION_VIEW);
208             intent.setDataAndType(Uri.fromFile(f),"audio/*");
209             startActivity(intent);
210         }
211     }
```

① 第 22 行至第 211 行定义一个类 SoundRecordExampleActivity 继承 Activity 类。

② 第 60 行至第 64 行判断 SD Card 是否插入，取得 SD Card 路径作为录音的文件位置。

③ 第 69 行至第 72 行创建 ArrayAdapter 类的对象 adapter，将 ArrayAdapter 添加到 ListView 对象中。

④ 第 75 行至第 105 行录音。第 86 行创建录制的音频文件，文件名的后缀为.amr，第 89 行至第 94 行设置录音的来源、格式、音频编码、音频文件保存路径，设置完成后开始录音，第 96 行至第 98 行设置图片按钮的可用状态。

⑤ 第 108 行至第 123 行停止录音。第 113 行至第 115 行如果正在录音，则停止录音并释放 MediaRecorder 对象，第 117 行至第 118 行从适配器对象 adapter 中取出当前音频文件名，并在 TextView 对象中显示停止信息和音频文件名。

⑥ 第 126 行至第 134 行播放录制的音频文件。第 129 行至第 131 行，如果文件存在，则播放音频文件，第 131 行打开指定的音频文件。

⑦ 第 136 行至第 154 行删除录制的音频文件。第 141 行删除 adapter 对象中指定的条目，第 145 行删除音频文件，第 148 行至第 151 行设置图片按钮的可用状态。

(5) 在根目录下的 AndroidManifest.xml 文件中，添加权限。

```
<?xml version = "1.0" encoding = "utf - 8"?>
< manifest xmlns:android = "http://schemas.android.com/apk/res/android"
    package = "com.application.soundrecordexample"
    android:versionCode = "1"
    android:versionName = "1.0.0">
    < application
      android:icon = "@drawable/ic_launcher"
      android:label = "@string/app_name">
      < activity android:name = "com.application.soundrecordexample.SoundRecordExampleActivity"
                  android:label = "@string/app_name">
           < intent - filter >
               < action
               android:name = "android.intent.action.MAIN" />
               < category
               android:name = "android.intent.category.LAUNCHER" />
           </intent - filter >
       </activity >
     </application >

     <! -- 添加录音权限 -->
     < uses - permission android:name = "android.permission.RECORD_AUDIO"/>

     <! -- 添加存储卡的可写入权限 -->
     < uses - permission android:name = "android.permission.WRITE_EXTERNAL_STORAGE"/>
</manifest >
```

【运行结果】

在 Eclipse 中启动模拟器，然后运行项目 SoundRecordExample，声音采集初始界面如图 8.16 所示，在第一行的按钮中，单击“录音”图片按钮，开始录音，单击“停止”图片按

钮，停止录音并显示录制的音频文件列表，如图 8.17 所示。选择列表中某一音频文件后，在第二行的按钮中，单击“播放”图片按钮即开始播放声音，单击“删除”图片按钮即可删除该文件。

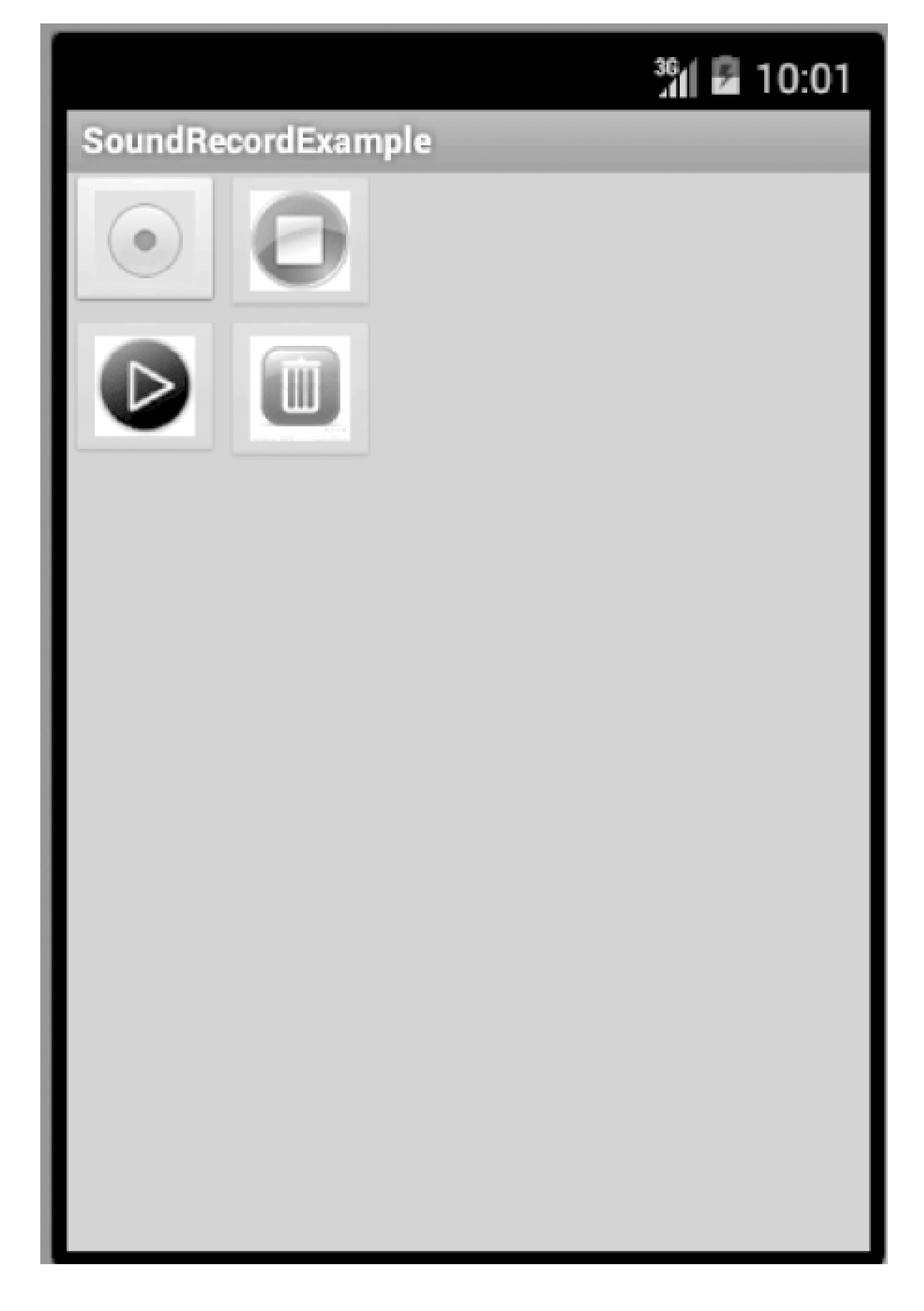


图 8.16　声音采集初始界面



图 8.17　已录制一个音频文件

8.5.2　图像采集

图像采集有两种方法：使用手机自带的 Camera 应用程序和使用 Camera 类获得摄像头信息。

使用 Camera 类时需要增加摄像头访问权限：

```
<uses-permission android:name="android.permission.CAMERA"/>
```

向 SD 卡中保存照片文件需要增加可写权限：

```
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
```

注意：使用模拟器不能显示摄像头的预览界面，在真机上可以显示。

【例 8.10】　图像采集举例。

【解题思路】

使用用户界面的按钮调用 Android 自带的 Camera 应用程序。拍照操作由 Camera 应用程序完成。拍照完成后向用户界面返回照片。

【开发步骤和程序分析】

(1) 在 Eclipse 中创建一个 CameraImageExample 应用项目，包名为 com. application.

cameraimageexample。

(2) 设计布局，定义的用户界面用于启动 Android 自带的 Camera 应用程序，并显示 Camera 应用程序拍摄的照片。当 Camera 应用程序退出时，返回初始用户界面。

在 res/layout 目录下的 main.xml 文件中，编辑代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
     android:orientation="vertical"
     android:layout_width="fill_parent"
     android:layout_height="fill_parent"
     android:gravity="center_horizontal|center_vertical">

          <!-- 定义相对布局 -->
          <RelativeLayout android:layout_width="fill_parent"
          android:layout_height="wrap_content"
          android:layout_weight="6"
          android:gravity="center_horizontal|center_vertical">

               <!--设置"无照片"文本框，位于相对布局的中央，由 Java 代码控制可见或隐蔽状态-->
               <TextView android:id="@+id/field"
                android:layout_width="fill_parent"
                android:layout_height="wrap_content"
                android:text="无照片"
                android:gravity="center_horizontal"/>

                <!--设置"照片"图片视图，位于相对布局的中央，由 Java 代码控制可见或隐蔽状态-->
               <ImageView android:id="@+id/image"
                 android:layout_width="fill_parent"
                 android:layout_height="wrap_content"/>
           </RelativeLayout>

           <!--设置"调用 Camera 程序"按钮-->
           <Button android:id="@+id/button"
                 android:layout_width="fill_parent"
                 android:layout_height="wrap_content"
                 android:text="调用 Camera 程序"
                 android:layout_weight="1"/>
</LinearLayout>
```

(3) 在 com.application.cameraimageexample 包下的 CameraImageExampleActivity.java 文件中，定义一个类 CameraImageExampleActivity 继承 Activity 类，重写 onCreate 方法，定义一个内部类 ButtonClickHandler 实现监听器接口，启动 Camera 程序，重写 onActivityResult()方法，显示捕获的照片，重写生命周期方法 onSaveInstanceState()，重写

生命周期方法 onRestoreInstanceState()。在该文件中编辑代码如下：

```
package com.application.cameraimageexample;

import java.io.File;
import com.application.cameraimageexample.R;
import android.app.Activity;
import android.content.Intent;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.net.Uri;
import android.os.Bundle;
import android.os.Environment;
import android.provider.MediaStore;
import android.view.View;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.TextView;

  //定义一个类 CameraImageExampleActivity 继承 Activity 类
 public class CameraImageExampleActivity extends Activity {
     protected Button u_button;
     protected ImageView u_image;
     protected TextView u_field;
     protected String u_path;
     protected boolean u_taken;
     protected static final String PHOTO_TAKEN = "photo_taken";

     //重写 onCreate 方法
     @Override
     public void onCreate(Bundle savedInstanceState) {
         super.onCreate(savedInstanceState);
         //设定屏幕显示为横向
         this.setRequestedOrientation(0);
                                //设置纵向显示为: setRequestedOrientation(90)
         setContentView(R.layout.main);

         u_image = ( ImageView ) findViewById( R.id.image );
         u_field = ( TextView ) findViewById( R.id.field );
         u_button = ( Button ) findViewById( R.id.button );
         u_button.setOnClickListener( new ButtonClickHandler() );
         //设置图片文件路径
         u_path = Environment.getExternalStorageDirectory() + "/take_picture.jpg";
     }

```

```
    public class ButtonClickHandler implements View.OnClickListener {
     public void onClick( View view ){
        File file = new File( u_path );
        Uri outputFileUri = Uri.fromFile( file );
        //使用 Android 自带的 Camera 捕捉图像程序
        Intent intent = new Intent(android.provider.MediaStore.ACTION_IMAGE_CAPTURE );
        intent.putExtra( MediaStore.EXTRA_OUTPUT, outputFileUri );
        startActivityForResult( intent, 0 );
     }
    }
    //重写 onActivityResult()方法
    @Override
    protected void onActivityResult(int requestCode, int resultCode, Intent data) {
        switch( resultCode ) {
            case 0:
                break;
            case -1:
                onPhotoTaken();
                break;
        }
    }
    //显示捕获的照片
    protected void onPhotoTaken() {
     u_taken = true;
     BitmapFactory.Options options = new BitmapFactory.Options();
     options.inSampleSize = 2;                //设置图片的容量大小为原始大小的 1/2
     Bitmap bitmap = BitmapFactory.decodeFile( u_path, options );
     u_image.setImageBitmap(bitmap);
     u_field.setVisibility( View.GONE );
    }

    //重写生命周期方法 onSaveInstanceState(),当系统要销毁 Activity 之前调用
    @Override
    protected void onSaveInstanceState( Bundle outState ) {
     outState.putBoolean( CameraImageExampleActivity.PHOTO_TAKEN, u_taken );
    }

    //重写生命周期方法 onRestoreInstanceState(),
    //当 Activity 被销毁后重新启用时被调用
    @Override
    protected void onRestoreInstanceState( Bundle savedInstanceState){
     if( savedInstanceState.getBoolean( CameraImageExampleActivity.PHOTO_TAKEN ) ) {
        onPhotoTaken();
     }
}
```

```
88 }
```

① 第 18 行至第 88 行定义一个类 CameraImageExampleActivity 继承 Activity 类。

② 第 28 行至第 41 行重写 onCreate 方法。

③ 第 43 行至第 52 行定义一个内部类 ButtonClickHandler 实现监听器接口,第 48 行创建一个 Intent 对象 intent,这个 intent 是 Android 自带的 Camera 捕捉图像程序,第 50 行启动 Camera 程序。

④ 第 54 行至第 63 行重写 onActivityResult()方法。

⑤ 第 65 行至第 72 行显示捕获的照片。

⑥ 第 75 行至第 78 重写生命周期方法 onSaveInstanceState(),当系统要销毁 Activity 之前调用,用于保存该 Activity 的状态。

⑦ 第 82 行至第 87 行重写生命周期方法 onRestoreInstanceState(),当 Activity 被销毁后重新启用时被调用,用于恢复该 Activity 的状态。

(4) 添加权限:

```
<?xml version = "1.0" encoding = "utf - 8"?>
<manifest xmlns:android = "http://schemas.android.com/apk/res/android"
    package = "com.application.cameraimageexample"
    android:versionCode = "1"
    android:versionName = "1.0" >
    <uses - sdk android:minSdkVersion = "10" />

    <application
        android:icon = "@drawable/ic_launcher"
         android:label = "@string/app_name" >
         <activity
         android:label = "@string/app_name"
         android:name = "com.application.cameraimageexample.CameraImageExampleActivity" >
         <intent - filter >
             <action android:name = "android.intent.action.MAIN" />
             <category android:name = "android.intent.category.LAUNCHER" />
         </intent - filter >
     </activity >
   </application >

   <! -- 添加存储卡的可写入权限 -->
   <uses - permission android:name = "android.permission.WRITE_EXTERNAL_STORAGE"/>
</manifest >
```

【运行结果】

在 Eclipse 中启动模拟器,然后运行项目 CameraImageExample,图像采集界面如图 8.18 所示。



图 8.18　图像采集界面

8.6　多媒体服务应用举例

综合应用本章和前面介绍的知识和技术，进行多媒体服务应用开发，举例如下。

【例 8.11】　音乐播放器举例。

【解题思路】

设计一个音乐播放器，对指定的歌曲进行播放、暂停、停止等控制，并显示歌词等信息，按下 Menu 键时，显示"退出"菜单项，并有退出确认对话框。

定义两个类，一个是继承 Activity 类的子类，用于显示布局和按钮对象和歌曲信息，并可与用户交互。一个是继承 Service 类的子类，用于对歌曲的播放、暂停、停止等进行控制。

在两个类中，分别定义广播接收器子类。Activity 子类中的广播接收器子类用于设置按钮的显示图片、状态变量的值，并广播用户的按键信息。Service 子类中的广播接收器子类用于控制歌曲的播放、暂停、停止等操作，并广播当前的播放状态。

【开发步骤和程序分析】

(1) 在 Eclipse 中创建一个 MusicPlayerExample 应用项目，包名为 com. application . musicplayerexample。

(2) 准备资源。

① 复制图片文件到 res\drawable-mdpi 下。

② 在 res 下创建子目录 raw，复制 MP3 文件到其下。

③ 设置需要使用的颜色：colors. xml。

④ 设置歌曲信息字符串：strings. xml。

(3) 设计布局，在界面上部，放置两个图片按钮和歌曲名称等信息，界面下部用滚动形式显示歌词信息，退出时有选项菜单和确认对话框，设置背景。

在 res/layout 目录下的 main. xml 文件中，编辑代码如下：

```
<?xml version = "1.0" encoding = "utf - 8"?>

<! -- 定义垂直分布的线性布局 -->
< LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = "vertical"
    android:background = "@drawable/back"
    android:layout_width = "fill_parent"
    android:layout_height = "fill_parent">
    < LinearLayout
            android:orientation = "horizontal"
            android:layout_width = "fill_parent"
            android:layout_height = "wrap_content">
            <! -- 设置图片按钮 -->
            < ImageButton
                  android:id = "@ + id/start"
                  android:layout_width = "wrap_content"
                  android:layout_height = "wrap_content"
                  android:src = "@drawable/playbt"/>
            < ImageButton
                  android:id = "@ + id/stop"
                  android:layout_width = "wrap_content"
                  android:layout_height = "wrap_content"
                  android:src = "@drawable/stopbt"/>
            < LinearLayout
                  android:orientation = "vertical"
                  android:layout_width = "fill_parent"
                  android:layout_height = "fill_parent">

                  <! -- 设置歌曲信息 ->
                  < TextView
                        android:id = "@ + id/textView1"
                        android:layout_width = "wrap_content"
                        android:layout_height = "wrap_content"
                        android:textSize = "20px"
                        android:textColor = "#ffffff"
                        android:ellipsize = "marquee"
                        android:layout_weight = "1"
                        android:marqueeRepeatLimit = "marquee_forever"
                        android:text = "@string/myTextView1"/>
                  < TextView
                        android:id = "@ + id/textView2"
                        android:textSize = "15px"
                        android:gravity = "center_vertical"
                        android:layout_weight = "1"
                        android:layout_width = "wrap_content"
```

```
                        android:layout_height = "wrap_content"
                        android:text = "@string/myTextView2"/>
                  < TextView
                        android:id = "@ + id/textView3"
                        android:textSize = "15px"
                        android:gravity = "center_vertical"
                        android:layout_weight = "1"
                        android:layout_width = "wrap_content"
                        android:layout_height = "wrap_content"
                        android:text = "@string/myTextView3"/>
            </LinearLayout >
        </LinearLayout >
        < ScrollView
              android:layout_width = "fill_parent"
              android:layout_height = "fill_parent"
              android:fillViewport = "true"
              >
              <! -- 设置列表视图 ListView,歌词信息布局 -- >
              < ListView
                  android:id = "@ + id/singtext"
                  android:layout_width = "fill_parent"
                  android:layout_height = "fill_parent"
              android:ellipsize = "marquee"
                  />
        </ScrollView >
   </LinearLayout >
```

(4) 在 com. application. musicplayerexample 包下的 MusicPlayerExampleActivity. java 文件中,定义一个类 MusicPlayerExampleActivity 继承 Activity 类,且实现事件监听器接口,重写的 onCreate()方法,创建 BaseAdapter 类的对象 myAdapter,为 ListView 类的对象 lv 添加适配器;定义一个内部类 ActivityReceiver 继承 BroadcastReceiver 类,进行播放状态判断,值为 1 表示没有声音播放,值为 2 表示正在播放,值为 3 表示暂停播放;重写 onClick()方法,重写 onDestroy()方法。在该文件中编辑代码如下:

```
package com.application.musicplayerexample;

import com.application.musicplayerexample.R;
import android.app.Activity;
import android.app.AlertDialog;
import android.app.Dialog;
import android.content.BroadcastReceiver;
import android.content.Context;
mport android.content.DialogInterface;
import android.content.Intent;
import android.content.IntentFilter;
```

```
import android.os.Bundle;
import android.view.Gravity;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.view.ViewGroup;
import android.view.View.OnClickListener;
import android.widget.BaseAdapter;
import android.widget.ImageButton;
import android.widget.LinearLayout;
import android.widget.ListView;
import android.widget.TextView;

  //定义一个类 MusicPlayerExampleActivity 继承 Activity 类,且实现事件监听器接口
  public class MusicPlayerExampleActivity extends Activity implements OnClickListener{
      ImageButton start;                              //播放、暂停按钮
      ImageButton stop;                               //停止按钮
      ActivityReceiver activityReceiver;
      int status = 1;                 //当前的状态,1 没有声音播放 ,2 正在播放声音,3 暂停
      ListView lv;

      //重写的 onCreate 方法
      @Override
      public void onCreate(Bundle savedInstanceState) {
          super.onCreate(savedInstanceState);

          //调用 setContentView()方法引用 main 布局
          setContentView(R.layout.main);
          lv = (ListView)findViewById(R.id.singtext); //获得 ListView 对象的引用
          BaseAdapter myAdapter = new BaseAdapter(){      //为 ListView 准备内容适配器
          //定义歌词字符串的数组
          String[] singtxts = getResources().getStringArray(R.array.singtexts);
          //歌词字符串的数组的总条目数,这里共 23 条
          public int getCount() {return singtxts.length;}
          public Object getItem(int arg0) { return null; }
          public long getItemId(int arg0) { return 0; }
          public View getView(int arg0, View arg1, ViewGroup arg2) {
              //动态生成每个下拉项对应的 View,每个下拉项 View 由 LinearLayout
              //中包含一个 TextView 构成
              LinearLayout ll = new LinearLayout(MusicPlayerExampleActivity.this);
              //初始化 LinearLayout
              ll.setOrientation(LinearLayout.HORIZONTAL);                //设置朝向
              ll.setPadding(3,0,0,0);                     //设置列表框的四周留白
              TextView tv = new TextView(MusicPlayerExampleActivity.this);
                                                          //初始化 TextView
```

```
            tv.setText(singtxts[arg0].toString());   //设置内容
            tv.setTextSize(15);                      //设置字体大小
            tv.setTextColor(MusicPlayerExampleActivity.this.getResources()
                    getColor(R.color.stxt));          //设置字体颜色
            tv.setPadding(3,0,0,0);                  //设置文本控件的四周留白
            tv.setGravity(Gravity.LEFT);             //设置文字居左对齐
            ll.addView(tv);                          //添加到 LinearLayout 中
            return ll;
        }
    };
        lv.setAdapter(myAdapter);
        lv.setVisibility(View.INVISIBLE);

        start = (ImageButton) this.findViewById(R.id.start);     //得到 start 的引用
        stop = (ImageButton) this.findViewById(R.id.stop);    //得到 stop 按钮的引用
        start.setOnClickListener(this);              //为按钮添加监听
        stop.setOnClickListener(this);               //为按钮添加监听
        activityReceiver = new ActivityReceiver();   //创建 BroadcastReceiver
        IntentFilter filter = new IntentFilter();    //创建 IntentFilter 过滤器
        filter.addAction("com.application.musicplayerexample.update"); //添加 Action
        registerReceiver(activityReceiver, filter);  //注册监听
        Intent intent = new Intent(this, MusicPlayerService.class);     //创建 Intent
        startService(intent);                        //启动后台 Service
    }

    /* 定义一个内部类 ActivityReceiver 继承 BroadcastReceiver 类 */
    public class ActivityReceiver extends BroadcastReceiver{
        @Override
        public void onReceive(Context context, Intent intent) {  //重写 onReceive 方法
            int mupdate = intent.getIntExtra("musicupdate", -1);
                                                     //得到 intent 中的数据
            switch(mupdate){                         //多分支判断
            case 1:                                  //没有声音播放
                start.setImageResource(R.drawable.playbt);          //更换图片
                status = 1;                          //设置当前状态
                break;
            case 2:                                  //正在播放声音
                start.setImageResource(R.drawable.pausebt);         //更换图片
                status = 2;                          //设置当前状态
                break;
            case 3:                                  //暂停中
                start.setImageResource(R.drawable.playbt);          //更换图片
                status = 3;                          //设置当前状态
                break;
            }
```

```
            }
        }

        /* 重写 onClick() 方法 */
        @Override
        public void onClick(View v) {
            Intent intent = new Intent("com.application.musicplayerexample.control");
                                                            //创建 Intent
            switch(v.getId()){                              //多分支判断
            case R.id.start:                                //按下播放、暂停按钮
                lv.setVisibility(View.VISIBLE);
                intent.putExtra("ACTION", 1);               //存放数据
                sendBroadcast(intent);                      //发送广播
                break;
            case R.id.stop:                                 //按下停止按钮
                intent.putExtra("ACTION", 2);               //存放数据
                sendBroadcast(intent);                      //发送广播
                break;
            }
        }

         /* 重写 onDestroy() 方法 */
        @Override
        protected void onDestroy() {                        //释放时被调用
            super.onDestroy();
            Intent intent = new Intent(this, MusicPlayerService.class);    //创建 Intent
            stopService(intent);                            //停止后台的 Service
        }

        /*退出菜单及对话框*/
        @Override
        public boolean onCreateOptionsMenu(Menu menu){      //弹出菜单
            menu.add(0,Menu.FIRST,0,"退出")
                .setIcon(android.R.drawable.ic_menu_delete);            //设置图标
            return true;
        }
        @Override
        public boolean onOptionsItemSelected(MenuItem item){          //选择的菜单项
            switch(item.getItemId()){                       //分支判断
            case Menu.FIRST:
                showDialog(1);                              //显示对话框
                break;
            }
            //将来可在此进行扩展
            return false;
```

```
144         }
145         @Override
146         protected Dialog onCreateDialog(int id){          //创建对话框
147             switch(id){                                   //判 断
148             case 1:
149                 return new AlertDialog.Builder(this)
150                     .setTitle("您确定退出?")
151                     .setPositiveButton("确定", new android.content.DialogInterface
152                                         .OnClickListener(){
153                         @Override
154                         public void onClick(DialogInterface dialog, int which) {
155                             System.exit(0);               //直接退出
156                         }
157                     })
158                     .setNegativeButton("取消", null)    //取消按钮
159                     .create();
160             default:
161                 return null;
162             }
163         }
164   }
```

① 第 26 行至第 164 行定义一个类 MusicPlayerExampleActivity 继承 Activity 类，且实现事件监听器接口。

② 第 33 行至第 79 行重写 onCreate()方法。

③ 第 41 行至第 65 行创建 BaseAdapter 类的对象 myAdapter，为 ListView 准备内容适配器，第 66 行为 ListView 类的对象 lv 添加适配器，第 67 行设置 lv 不可见。

④ 第 73 行至第 76 行创建并动态注册广播接收器 activityReceiver，第 77 行至第 78 行启动后台服务，其服务定义在子类 MusicService 中。

⑤ 第 81 行至第 101 行定义一个内部类 ActivityReceiver 继承 BroadcastReceiver 类。第 85 行获取 intent 中的数据，第 86 行至第 99 行进行播放状态判断，值为 1 表示没有声音播放，值为 2 表示正在播放，值为 3 表示暂停播放。

⑥ 第 104 行至第 118 行重写 onClick()方法。第 106 行创建 Intent 对象，第 107 行至第 117 行，当按下播放/暂停按钮或停止按钮时，存放不同的数据和发送不同的广播信息。

⑦ 第 121 行至第 126 行重写 onDestroy()方法，在该方法中停止服务。第 130 行至第 134 行创建选项菜单，第 145 行至第 163 行创建对话框。

(5) 在 com.application.musicplayerexample 包下的 MusicPlayerService 文件中，定义一个类 MusicPlayerService 继承 Service 类，重写 onCreate()方法，创建 BroadcastReceiver 对象，注册 BroadcastReceiver；重写 onDestroy()方法，在该方法中注销 BroadcastReceiver 对象；定义内部类 ServiceReceiver 继承 BroadcastReceiver 类，重写 onReceive()方法，根据

action 和 status 的值进行处理。在该文件中编辑代码如下：

```
package com.application.musicplayerexample;

import com.application.musicplayerexample.R;
import android.app.Service;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.content.IntentFilter;
import android.media.MediaPlayer;
import android.os.IBinder;

   //定义一个类 MusicPlayerService 继承 Service 类
   public class MusicPlayerService extends Service{
       MediaPlayer mp;
       ServiceReceiver serviceReceiver;
       int status = 1;              //当前的状态,1 没有声音播放 ,2 正在播放声音,3 暂停
       @Override
       public IBinder onBind(Intent intent) {              //重写的 onBind 方法
           return null;
       }

       //重写 onCreate() 方法
       @Override
       public void onCreate() {
           status = 1;
           serviceReceiver = new ServiceReceiver();     //创建 BroadcastReceiver
           IntentFilter filter = new IntentFilter();    //创建过滤器
           filter.addAction("com.application.musicplayerexample.control");
                                                                        //添加 Action
           registerReceiver(serviceReceiver, filter);   //注册 BroadcastReceiver
           super.onCreate();
       }

       //重写 onDestroy() 方法
       @Override
       public void onDestroy() {
           unregisterReceiver(serviceReceiver);         //取消注册
           super.onDestroy();
       }

       //定义内部类 ServiceReceiver 继承 BroadcastReceiver 类
       public class ServiceReceiver extends BroadcastReceiver{
           @Override
```

```
43            public void onReceive(Context context, Intent intent) {          //重写响应方法
44                int action = intent.getIntExtra("ACTION", -1);          //获取需要的数据
45                switch(action){
46                case 1:  //播放或暂停声音
47                    if(status == 1){                      //当前没有声音播放
48                        mp = MediaPlayer.create(context, R.raw.tibetanplateau);
49                        status = 2;
50                        Intent sendIntent = new
51                                Intent("com.application.musicplayerexample.update");
52                        sendIntent.putExtra("musicupdate", 2);
53                        sendBroadcast(sendIntent);
54                        mp.start();
55                    }
56                    else if(status == 2){                 //正在播放声音
57                        mp.pause();                       //停止
58                        status = 3;                       //改变状态
59                        Intent sendIntent = new
60                                Intent("com.application.musicplayerexample.update");
61                        sendIntent.putExtra("musicupdate", 3);             //存放数据
62                        sendBroadcast(sendIntent);        //发送广播
63                    }
64                    else if(status == 3){                 //暂停中
65                        mp.start();                       //播放声音
66                        status = 2;                       //改变状态
67                        Intent sendIntent = new
68                                Intent("com.application.musicplayerexample.update");
69                        sendIntent.putExtra("musicupdate", 2);             //存放数据
70                        sendBroadcast(sendIntent);        //发送广播
71                    }
72                    break;
73                case 2:  //停止声音
74                    if(status == 2 || status == 3){       //播放中或暂停中
75                        mp.stop();                        //停止播放
76                        status = 1;                       //改变状态
77                        Intent sendIntent = new
78                                Intent("com.application.musicplayerexample.update");
79                        sendIntent.putExtra("musicupdate", 1);             //存放数据
80                        sendBroadcast(sendIntent);        //发送广播
81                    }
82                }
83            }
84        }
85    }
```

① 第 13 行至第 69 行定义一个类 MusicPlayerService 继承 Service 类。

② 第 23 行至第 31 行重写 onCreate()方法。第 26 行创建 BroadcastReceiver 对象，第 27 行创建过滤器，第 28 行添加 Action，第 29 行注册 BroadcastReceiver。

③ 第 34 行至第 38 行重写 onDestroy()方法，在该方法中注销 BroadcastReceiver 对象。

④ 第 40 行至第 84 行定义内部类 ServiceReceiver 继承 BroadcastReceiver 类，第 42 行至第 83 行重写 onReceive()方法，第 45 行至第 82 行根据 action 和 status 的值进行处理。

【运行结果】

在 Eclipse 中启动模拟器，然后运行项目 MusicPlayerExample，音乐播放器初始界面如图 8.19 所示。当单击“播放”按钮时，进行歌曲播放并显示歌词等信息，如图 8.20 所示。

按下 Menu 键显示“退出”菜单项，如图 8.21 所示。单击“退出”菜单项，弹出“确认”对话框，单击“确定”按钮即退出。



图 8.19　初始界面



图 8.20　播放歌曲



图 8.21　按下 Menu 键显示“退出”菜单项

8.7 小　　结

本章主要介绍了以下内容：

(1) 在 Android 中需要通过 Graphics 类来显示 2D 图形，Android 在 android.graphics 包内提供了 2D 图形库，常用类有 Color 类、Paint 类、Canvas 类、Path 类等。绘制 2D 图形包括绘制像素点、绘制直线、绘制圆形、绘制弧、绘制矩形、绘制椭圆、绘制路径、绘制文本等内容。

(2) 3D 图形的最基本的单位是顶点(Vertex)，它代表空间中的一个点，由若干点可以构成多边形，再由多边形构成复杂的空间图形。OpenGL ES 支持的基本图形只有点(Point)、线(Line)和三角形(Triangle)，所有 3D 图形都可以通过上述基本图形组合而成。

OpenGL ES 提供两类绘制 3D 图形的方法：glDrawArrays()方法和 glDrawElements()方法。

(3) 通常将动画分为逐帧动画(Frame Animation)和补间动画(Tween Animation)。

逐帧动画通过连续地播放一系列的图片文件，形成动画，它的三个要素是多幅图片、播放顺序和持续时间，逐帧动画用到的类 AnimationDrawable 包含在 android.graphics.drawable.AnimationDrawable 包下。

补间动画通过一系列的指令，将一个 View 对象进行位置、尺寸、旋转、透明度等变换从而形成动画，View 对象可以是图片、文本、按钮等对象。

(4) Android 播放音频的方式有两种：使用 MediaPlayer 类和使用 SoundPool 类。使用 MediaPlayer 类用于播放短促、反应速度快的声音，例如游戏中的音效配音，而 SoundPool 常用于播放较长的、对反应时间要求不高的声音，例如播放后台音乐、歌曲等。

Android 的音频文件存放在项目的 res/raw 文件夹下，Android 支持的音频格式有 MP3、MID、WAV、OGG、AMR 等，音频格式采样率为 11kHz、22kHz、44.1kHz，16 位立体声。

(5) 可以使用 MediaPlayer 和 SurfaceView 播放视频，也可以使用 VideoView 播放视频。Android 支持的视频格式有 MP4、3GP、H.263、H.264(AVC)等，比较大的视频文件常存放在 SD 卡中。

使用 MediaPlayer 和 SurfaceView 播放视频需要为 MediaPlayer 播放器创建一个用于绘制图像的控件 Surface，采用 MediaPlayer 与 SurfaceView 结合用来实现视频的输出。

使用 VideoView 播放视频，在 XML 文件中加入 VideoView 控件，再从 SD Card 中载入 MP4 文件或 3GP 文件，就能使用 VideoView 播放视频。

(6) 声音采集通过 MediaRecorder 类来实现，MediaRecorder 包含了 Audio 和 video 的记录功能，录制语音文件通常用.amr 格式，在 AndroidManifest.xml 中要添加录音等权限。

图像采集有两种方法：使用手机自带的 Camera 应用程序和使用 Camera 类获得摄像头信息。

习　题　8

一、选择题

8.1　绘制路径时，需要用到________类。

A. Paths　　B. Path　　C. Color　　D. Bitmsp

8.2　实现透明度补间动画时，使用________属性可指定动画的持续时间。

A. android:fromDegrees　　B. android:repeatCount

C. android:repeatMode　　D. android:duration

8.3　使用 MediaPlayer 播放音频时，调用该对象的________方法可停止音频。

A. pause()　　B. start()　　C. stop()　　D. play()

二、填空题

8.4　Android 中需要通过________类来显示 2D 图形。

8.5　绘制 2D 图形常用类有 Color 类、________类、Canvas 类、Path 类等。

8.6　OpenGL ES 支持的基本图形只有点、线和________。

8.7　逐帧动画通过连续地播放________的图片文件形成动画。

8.8　补间动画通过一系列的指令，将一个 View 对象进行位置、尺寸、________、透明度等变换从而形成动画。

8.9　Android 播放音频的方式有两种：使用________类和使用 SoundPool 类。

8.10　Android 使用 MediaPlayer 和 SurfaceView 播放视频，也可以使用________播放视频。

8.11　声音采集通过________类来实现，录制声音文件通常用.amr 格式。

三、问答题

8.12　Path 类常用的方法有哪些？各有何功能？

8.13　绘制直线、圆、矩形分别要调用哪些方法？

8.14　简述使用 MediaPlayer 类播放音频的步骤。

8.15　简述使用 SoundPool 类播放音频的步骤。

8.16　简述使用 MediaPlayer 和 SurfaceView 播放视频的步骤。

8.17　简述使用 VideoView 播放视频的步骤。

四、应用题

8.18　制作一个循环播放花卉的逐帧动画，具有“显示”和“停止”按钮。

8.19　制作一个循环播放风景的补间动画，具有“显示”和“停止”按钮。

8.20　设计一个音频播放器，使用 MediaPlayer 类，具有音频播放和暂停等功能。

8.21　设计一个音频播放器，使用 SoundPool 类，具有音频播放和暂停等功能。

8.22　设计一个视频播放器，使用 MediaPlayer 和 SurfaceView 类，具有视频播放和暂停等功能。

8.23　设计一个视频播放器，使用 VideoView 类，具有视频播放和暂停等功能。

8.24　设计一个录音器，使用 MediaRecorder 类，具有录音、停止、播放、删除等功能。

8.25　设计一个摄像头，使用自带的 Camera 应用程序，具有拍照功能。

8.26　设计一个音乐播放器，具有歌曲播放、暂停、停止等功能，并显示歌词等信息。

第9章 定位服务和百度地图应用开发

本章要点

- LBS(Location Based Service)被称为定位服务或基于位置的服务,提供与空间位置相关的综合应用服务。
- LBS服务模式有:生活信息服务模式、电子商务模式、社交网络和游戏模式等。
- Android定位方式一般有以下几种:GPS、NLP(Android Network Location Provider,网络提供者)、第三方提供的地图应用接口。
- 百度地图Android SDK是一套基于Android 2.3及以上版本设备的应用程序接口。应用开发人员可以使用该套SDK开发适用于Android系统移动设备的地图应用。

LBS(Location Based Service)定位服务,融合了GPS(Global Positioning System)定位、移动通信、导航等多种技术,从团购、导航到社交等,近几年来发展非常迅速。本章介绍定位服务概述、获取位置信息、百度地图应用开发等内容。

9.1 定位服务概述

本节介绍LBS简介和LBS服务模式等内容。

9.1.1 LBS简介

LBS(Location Based Service)被称为定位服务或基于位置的服务,融合了GPS定位、移动通信、导航等多种技术,提供与空间位置相关的综合应用服务,当前流行的手机应用软件,如百度地图、滴滴打车、美团等,都采用了LBS位置服务。

LBS包括两层含义:定位和服务。LBS首先是确定移动设备所在的地理位置,可以使用定位软件,如Google地图、百度地图等。定位服务需要在互联网或无线网络的环境中进行,并且使用到地理信息系统GIS(Geographic Information System)。基于位置的服务发展十分迅速,包括商务、医疗、工作和生活的各个方面,为用户提供定位、追踪和敏感区域警告等一系列服务。

9.1.2 LBS服务模式

LBS服务模式有:生活信息服务模式、电子商务模式、社交网络和游戏模式等,下面分别介绍。

1. 生活信息服务模式

以生活信息类网站或者点评网与地理位置服务结合的模式，为用户提供餐饮、旅游、娱乐、票务、休闲等信息，该模式通过 LBS 实现生活信息服务的增值，同时为用户提供更准确、更方便的信息指南，也为商户提供一个基于位置的精准营销平台，其代表有美团、大众点评网等，例如，美团网为消费者发现最值得信赖的商家，让消费者享受低折扣的优质服务，为商家找到最合适的消费者，给商家提供最大收益的互联网推广，美团定位服务如图 9.1 所示。

2. 电子商务模式

LBS 和电子商务合作的模式，目前最流行的是滴滴打车，滴滴打车是涵盖出租车、专车、快车、顺风车多项业务在内的网约车出行平台。网约车有以下优点：将用户和出租车、私家车等连接起来，更有效地利用资源，降低用户出行成本，提高出行效率，方便偏远地区用户的出行，提供更多的就业机会，助力于公务车改革，加速出租车互联网化等。滴滴打车定位服务如图 9.2 所示。




图 9.1　美团定位服务



图 9.2　滴滴打车定位服务

3. 社交网络和游戏模式

该模式最初需要用户主动签到以记录自己所在的位置，随着移动互联网的发展，已被微

信、微博等具有广泛用户基础的社交应用所取代，这些软件在它们身上所嵌入的 LBS 功能，比单纯签到软件更具市场价值，例如，微信中的“附近的人”，方便用户查找附近的社交对象。

9.2 获取位置信息

Android 定位方式一般有以下几种：GPS、NLP(Android Network Location Provider，网络提供者)、第三方提供的地图应用接口。

1. GPS

GPS 定位的优点是定位信息比较精确，平均精度在 10m 左右；缺点则是信息返回较慢，定位时间往往在几十秒到几分钟不等，且只能在户外使用，耗电严重。

2. NLP

NLP 通过基站和 Wi-Fi 信号获取位置信息，优点是室内外均可使用，耗电少，定位时间短，一般只需几秒；缺点是定位不够精确。

3. 第三方提供的地图应用接口

现阶段的 Android LBS 开发通常使用第三方提供的地图应用接口。这些地图应用可以通过 SDK 嵌入到程序中，常用的有 Google 地图、百度地图和高德地图等。

9.3 百度地图应用开发

百度地图 Android SDK 是一套基于 Android 2.3 及以上版本设备的应用程序接口。应用开发人员可以使用该套 SDK 开发适用于 Android 系统移动设备的地图应用，通过调用地图 SDK 接口，可以轻松访问百度地图服务和数据，构建功能丰富、交互性强的地图类应用程序。百度地图 Android SDK 提供的所有服务是免费的，接口使用无次数限制，但必须申请密钥(key)后，才可使用百度地图 Android SDK。

百度地图 Android SDK 具有以下功能。

(1) 地图：提供地图展示和地图操作功能。

(2) POI 检索：支持周边检索、区域检索、城市内检索和 Place 详情信息检索。

(3) 地理编码：提供地理坐标和地址之间相互转换的能力。

(4) 线路规划：支持公交信息查询、公交换乘查询、公交/驾车/骑行/步行线路规划。

(5) 地图覆盖物：百度地图 SDK 支持多种地图覆盖物，帮助展示更丰富的地图。

(6) 定位：采用 GPS、WI-FI、基站、IP 混合定位模式。

(7) 离线地图：使用离线地图可节省用户流量，提供更好的地图展示效果。

(8) 调启百度地图：利用 SDK 接口，直接在本地打开百度地图客户端或 WebApp，实现地图功能。目前支持调启的功能有：POI 周边检索、POI 详情页面、步行线路规划、驾车线路规划、公交线路规划、驾车导航、步行导航、骑行导航。

(9) 周边雷达：周边雷达功能，是面向移动端开发者的一套 SDK 功能接口。

(10) LBS 云：百度地图 LBS 云是百度地图针对 LBS 开发者全新推出的平台级服务，不仅适用 PC 应用开发，同时适用移动设备应用的开发。

(11) 特色功能：包括短串分享、Place 详情信息检索、热力图等。

(12) 个性化地图：自 v3.7.0 起，支持使用个性化地图模板，改变底图颜色和样式。

(13) 室内图：自 v4.0 起，百度地图 SDK 室内图功能正式上线，辅助开发者实现全新的地理位置服务体验，室内地图与百度地图 APP 同步更新。

(14) Android Wear：自 v4.0 起，适配 Android Wear，支持 Android 穿戴设备。

9.3.1 登录百度地图开发平台

在地址栏输入网址 http://lbsyun.baidu.com/，即可登录百度地图开发平台网站，如图 9.3 所示。



图 9.3 百度地图开发平台网站

在图 9.3 中，在“Android 开发”下拉菜单中选择“基础地图”，进入“Android 地图 SDK”，如图 9.4 所示。

9.3.2 申请应用开发密钥

进行百度地图应用开发，必须申请密钥后，才能使用百度地图 Android SDK。

申请应用开发密钥步骤如下：

1. 开发者注册

在图 9.4 中，单击左边的“获取密钥”选项，进入“百度地图开发平台开发者注册”页，如图 9.5 所示。

为进行开发者注册，必须首先注册一个百度账户。

在图 9.5 中，在“姓名”栏中输入注册的百度账户，在“手机”栏中输入手机号；在“邮箱”



图 9.4　Android 地图 SDK

栏中输入邮箱，获取验证码并进行验证后，单击“提交”按钮，在收到的开发者激活邮件中，单击邮件提供的链接，完成开发者账户激活。



图 9.5　开发者注册

2. 创建应用开发密钥

每个密钥(Key)唯一对应一个应用开发项目(APP)。

下面以包名为 com. application. baidumap 的应用项目 BaiduMap 为例介绍创建应用开发密钥。

开发者注册后，会跳转到 API 控制台服务，单击“创建应用”按钮，进入“创建应用”页，如图 9.6 所示。

在图 9.7 中，在“应用名称”栏中输入“BaiduMap”，在“应用类型”栏下拉菜单中选择



图 9.6　创建应用

Android SDK，在“发布版 SHA1”栏中输入 SHA1 值，在“包名”栏中输入“com.application.baidumap”，“安全码”栏中的值在输入 SHA1 值和包名后自动生成，其组成规则为：SHA1 值+“;”+包名，单击“提交”按钮。

为查看 Android 签名证书的 SHA1(数字签名)值，在 Eclipse 中，选择 Windows→Preferences→Android→Build，即可进行查看，如图 9.7 所示。

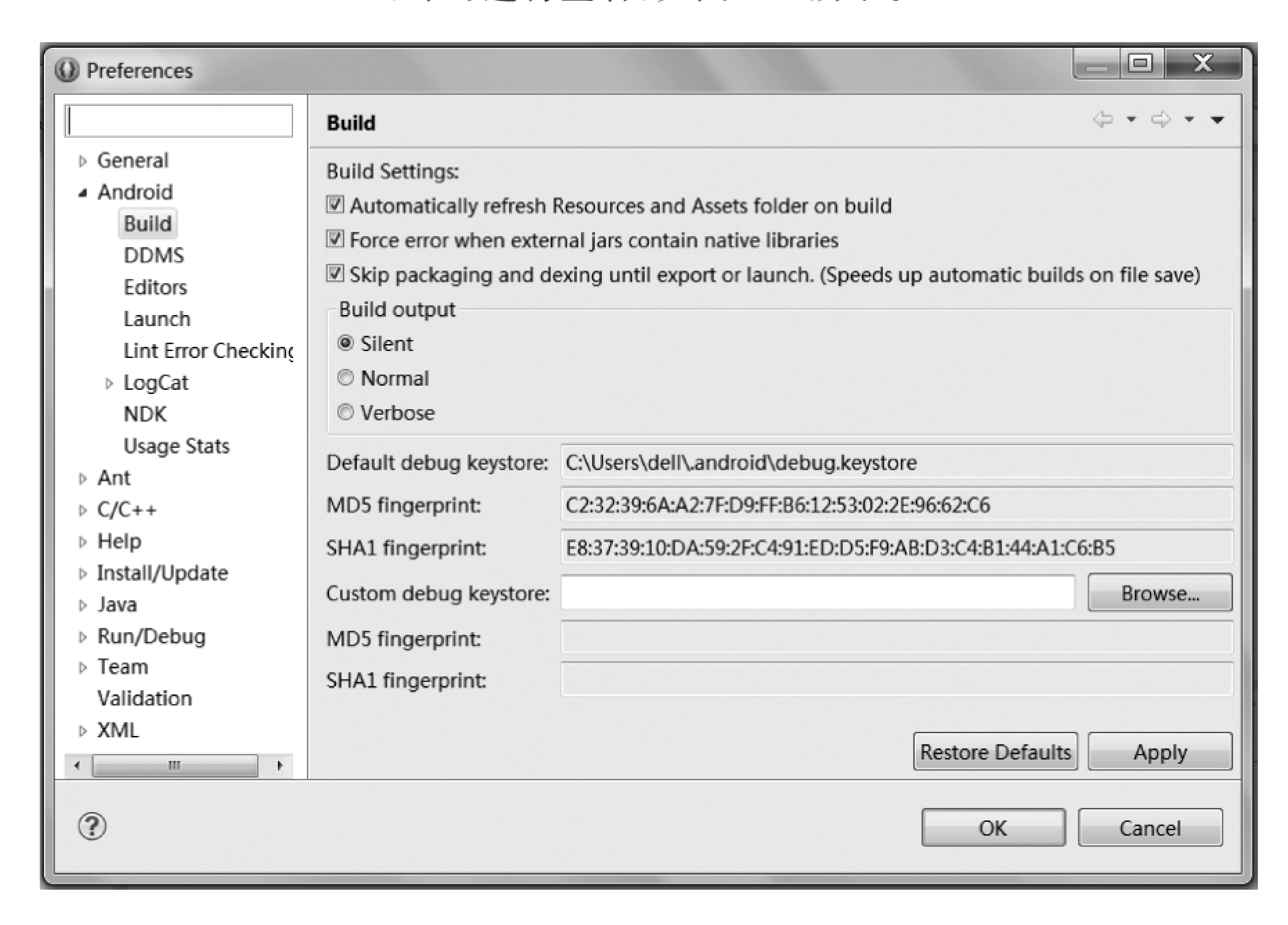


图 9.7　查看 SHA1 值

3. 查看应用开发密钥

单击“查看应用”按钮，进入“查看应用”页，即可查看创建的应用开发密钥（AK），如图 9.8 所示。



图 9.8 查看应用开发密钥

9.3.3 下载 SDK

在图 9.4 中，单击左边的“相关下载”选项，进入“相关下载. Android 地图 SDK”页，如图 9.9 所示。



图 9.9 进入相关下载

在图 9.9 中，单击“自定义下载”按钮，进入“开发资源下载平台”页，如图 9.10 所示。

用户可结合自身需求、自定义选择业务功能，打包下载所选功能开发包，本例选举了 8 项功能，单击“开发包”按钮，进行下载。



图 9.10 SDK 下载界面

9.3.4 开发 LBS 应用

下面以开发一个简单地图为例介绍开发 LBS 应用的步骤。

【例 9.1】 简单地图应用开发举例。

【解题思路】

本例为显示百度地图，需要在应用项目中使用百度地图数据的接口，并将申请的应用开发密钥添加到应用项目的 AndroidManifest.xml 文件中和配置相应的权限。

【开发步骤和程序分析】

(1) 在 Eclipse 中创建一个 BaiduMap 应用项目，包名为 com.application.baidumap。

(2) 将下载的 BaiduLBS_AndroidSDK_Lib.zip 压缩包解压，解压后将开发包 libs 中的文件复制到应用项目 BaiduMap 的 libs 文件夹下，如图 9.11 所示。



图 9.11 将开发包 libs 中的文件复制到应用项目的 libs 文件夹下

(3) 设计布局，地图界面用地图控件 MapView 提供，将其布局在相对布局中。

在 res/layout 目录下的 activity_main.xml

文件中，编辑代码如下：

```
<! -- 定义相对布局 -->
<RelativeLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    xmlns:tools = "http://schemas.android.com/tools"
    android:layout_width = "match_parent"
    android:layout_height = "match_parent"
    android:paddingBottom = "@dimen/activity_vertical_margin"
    android:paddingLeft = "@dimen/activity_horizontal_margin"
    android:paddingRight = "@dimen/activity_horizontal_margin"
    android:paddingTop = "@dimen/activity_vertical_margin"
        tools:context = ".MainActivity" >

        <! -- 定义 MapView 控件 -->
            <com.baidu.mapapi.map.MapView
            android:id = "@ + id/bmapView"
            android:layout_width = "fill_parent"
            android:layout_height = "fill_parent"
            android:clickable = "true" />
</RelativeLayout>
```

(4) 创建地图 Activity，在 Activity 中调用 MapView 控件，并在 Activity 生命周期中通过回调方法管理地图生命周期。

在 com.application.baidumap 包下的 MainActivity.java 文件中，编辑代码如下：

```
package com.application.baidumap;

import com.baidu.mapapi.SDKInitializer;
import com.baidu.mapapi.map.MapView;
import com.application.baidumap.R;
import android.os.Bundle;
import android.app.Activity;

//定义一个类 MainActivity 继承 Activity 类
public class MainActivity extends Activity {

        MapView mMapView;

        //重写 onCreate()方法
        @Override
        protected void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            SDKInitializer.initialize(getApplicationContext());
                                                    //初始化 SDK 各个功能组件
            setContentView(R.layout.activity_main);
            mMapView = (MapView) findViewById(R.id.bmapView);
```

```
21          }
22
23          //重写 onResume()方法
24          @Override
25          protected void onResume() {
26              //在 Activity 执行 onResume 时执行 mMapView. onResume (),实现地图生命周期管理
27              mMapView.onResume();
28              super.onResume();
29          }
30
31          //重写 onPause() 方法
32          @Override
33          protected void onPause() {
34              //在 Activity 执行 onPause 时执行 mMapView. onPause (),实现地图生命周期管理
35              mMapView.onPause();
36              super.onPause();
37          }
38
39          //重写 onDestroy()方法
40          @Override
41          protected void onDestroy() {
42              //在 Activity 执行 onDestroy 时执行 mMapView.onDestroy(),实现地图生命周期管理
43              mMapView.onDestroy();
44              super.onDestroy();
45          }
46  }
```

① 第 10 行至第 46 行定义一个类 MainActivity 继承 Activity 类。

② 第 15 行至第 21 行重写 onCreate()方法,第 18 行在使用 SDK 各组件之前初始化 context 信息,传入 ApplicationContext,第 19 行调用 setContentView()方法引用 activity_main 布局,第 20 行获取 MapView 控件的唯一标识。

③ 第 24 行至第 29 行重写 onResume()方法,在 Activity 执行 onResume()时执行 mMapView. onResume (),实现地图生命周期管理。

④ 第 32 行至第 37 行重写 onPause()方法,在 Activity 执行 onPause()时执行 mMapView. onPause(),实现地图生命周期管理。

⑤ 第 40 行至第 45 行重写 onDestroy()方法,在 Activity 执行 onDestroy()时执行 mMapView. onDestroy(),实现地图生命周期管理。

(5) 将应用开发密钥添加到应用项目的 AndroidManifest. xml 文件中,加入所需要的应用权限,以调用相应的接口和控件。

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
          package="com.application.baidumap"
          android:versionCode="1"
```

```
        android:versionName="1.0" >
        <!-- 添加所需权限 -->
        <uses-permission android:name="android.permission.GET_ACCOUNTS" />
        <uses-permission android:name="android.permission.USE_CREDENTIALS" />
        <uses-permission android:name="android.permission.MANAGE_ACCOUNTS" />
        <uses-permission android:name="android.permission.AUTHENTICATE_ACCOUNTS" />
        <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
        <uses-permission android:name="android.permission.INTERNET" />
        <uses-permission android:name="com.android.launcher.permission.READ_SETTINGS" />
        <uses-permission android:name="android.permission.CHANGE_WIFI_STATE" />
        <uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
        <uses-permission android:name="android.permission.READ_PHONE_STATE" />
        <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
        <uses-permission android:name="android.permission.BROADCAST_STICKY" />
        <uses-permission android:name="android.permission.WRITE_SETTINGS" />
        <uses-permission android:name="android.permission.READ_PHONE_STATE" />

        <uses-sdk
            android:minSdkVersion="8"
            android:targetSdkVersion="18" />

       <application
            android:allowBackup="true"
            android:icon="@drawable/ic_launcher"
            android:label="@string/app_name"
            android:theme="@style/AppTheme" >
            <!-- 添加开发密钥 -->
            <meta-data
                android:name="com.baidu.lbsapi.API_KEY"
                android:value="4ylHeINl8HOfOv6ruUHVaTAAhDslbho7" />
            <activity
                android:name="com.application.baidumap.MainActivity"
                android:label="@string/app_name" >
                <intent-filter>
                    <action android:name="android.intent.action.MAIN" />

                    <category android:name="android.intent.category.LAUNCHER" />
                </intent-filter>
            </activity>
       </application>

</manifest>
```

【运行结果】

在 Eclipse 中启动模拟器，然后运行项目 BaiduMap，运行结果如图 9.12 所示。

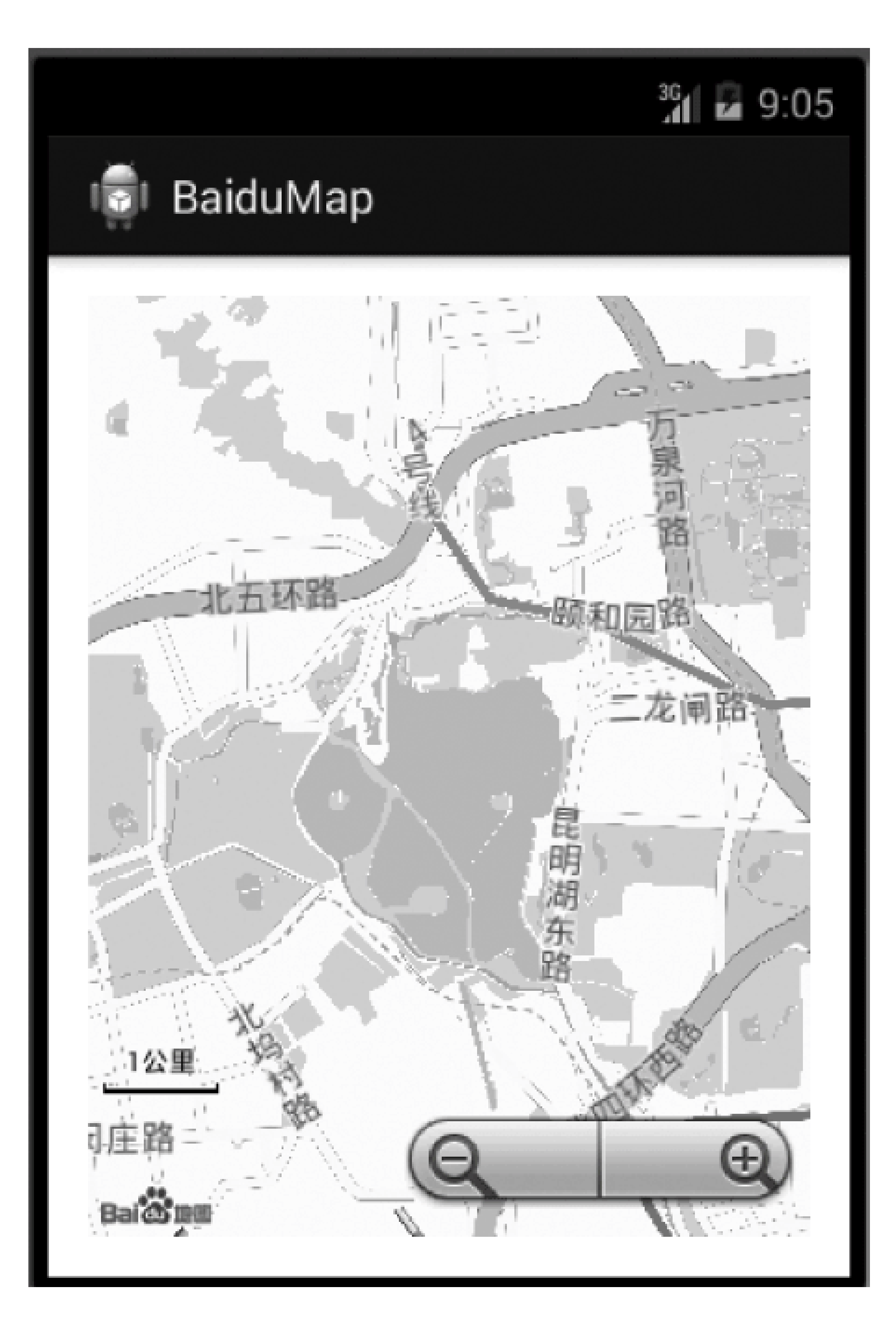


图 9.12　基本地图

9.4　小　　结

本章主要介绍了以下内容：

(1) LBS(Location Based Service)被称为定位服务或基于位置的服务，融合了 GPS 定位、移动通信、导航等多种技术，提供与空间位置和相关的综合应用服务，当前流行的手机应用软件，如百度地图、滴滴打车、美团等，都采用了 LBS 位置服务。

LBS 包括两层含义：定位和服务。LBS 首先是确定移动设备所在的地理位置，可以使用定位软件，如 Google 地图、百度地图等。定位服务需要在互联网或无线网络的环境中进行，并且使用到地理信息系统 GIS(Geographic Information System)。基于位置的服务发展十分迅速，包括商务、医疗、工作和生活的各个方面，为用户提供定位、追踪和敏感区域警告等一系列服务。

(2) LBS 服务模式有：生活信息服务模式、电子商务模式、社交网络和游戏模式等。

生活信息服务模式是生活信息类网站或者点评网与地理位置服务结合的模式，为用户提供餐饮、旅游、娱乐、票务、休闲等信息。

电子商务模式是 LBS 和电子商务合作模式，目前最流行的是滴滴打车。

社交网络和游戏模式最初需要用户主动签到以记录自己所在的位置，随着移动互联网的发展，已被微信、微博等具有广泛用户基础的社交应用所取代。

(3) Android 定位方式一般有以下几种：GPS、NLP(Android Network Location Provider，

网络提供者)、第三方提供的地图应用接口。

(4) 百度地图 Android SDK 是一套基于 Android 2.3 及以上版本设备的应用程序接口。应用开发人员可以使用该套 SDK 开发适用于 Android 系统移动设备的地图应用。

登录百度地图开发平台网站,申请应用开发密钥,下载百度地图 Android SDK,即可进入有关百度地图项目的应用开发。

习 题 9

一、选择题

9.1 LBS 服务模式不包括________。

A. 社交网络和游戏模式　　B. 生活信息服务模式

C. 概念模式　　D. 电子商务模式

9.2 Android 定位方式不包括________。

A. 第三方提供的地图应用接口　　B. NLP

C. GPS　　D. JSP

二、填空题

9.3 LBS(Location Based Service)被称为定位服务或________。

9.4 LBS 包括两层含义:________和服务。

9.5 LBS 服务模式有:生活信息服务模式、________、社交网络和游戏模式等。

9.6 生活信息服务模式是生活信息类网站或者点评网与________结合的模式。

9.7 Android 定位方式一般有以下几种:GPS、NLP、________。

9.8 百度地图 Android SDK 是一套基于 Android 2.3 及以上版本设备的应用程序________。

9.9 进行百度地图应用开发,必须________后,才能使用百度地图 Android SDK。

9.10 在百度地图应用开发中,每个密钥唯一对应一个________。

三、问答题

9.11 简述 LBS 的含义。

9.12 LBS 服务模式有哪些?各个模式有何特点?

9.13 Android 定位方式有哪几种?各种定位方式有何特点?

9.14 简述申请应用开发密钥的步骤。

9.15 简述有关百度地图项目的应用开发步骤。

四、应用题

9.16 参考例 9.1,开发一个简单的百度地图。

9.17 在上例的基础上,增加定位功能。

第 10 章　Android 应用项目开发

本章要点

- 网上求职手机客户端系统的需求分析和设计包括需求分析、系统目标、技术特点、系统模块结构图和表结构设计等内容。
- 网上求职手机客户端系统程序结构包含以下 4 类文件：Activity 类和 Fragment 类文件，Adapter 类和公共数据类文件，布局文件和其他资源文件。
- 网上求职手机客户端系统采用 Fragment 划分用户界面，将 Activity 提供的用户界面划分为上下两个部分，在 Activity 运行时，Fragment 将嵌入到用户界面的上部。
- 基本页面是用户界面中的重要部分，基本页面由首页、消息页、我的页组成。
- 如果用户尚未登录，当“消息”单选按钮被选中时，进入用户登录页，此时如果用户是首次登录，单击右下方的“注册”按钮，进入用户注册页。
- 进入首页，选择“兼职”文本框或“全职”文本框，单击某一职位条目，进入该职位的职位详情页。
- 进入我的页，如果用户已登录，单击“个人简历”文本框，进入个人简历页，单击“编辑资料”文本框，进入编辑资料页。

本章介绍的 Android 应用项目开发实例，是网上求职系统的组成部分，网上求职系统包括网上求职服务器端系统、网上求职手机客户端系统、网上求职 PC 客户端系统，本章仅介绍网上求职手机客户端系统，并对与服务器端的接口做了本地化处理。本章介绍网上求职手机客户端系统需求分析和设计、网上求职手机客户端系统程序结构设计、基本页面、用户登录和注册、职位详情、我的信息等内容。

10.1　网上求职手机客户端系统需求分析和设计

本节介绍需求分析、总体设计、数据库设计等内容。

10.1.1　需求分析

网上求职手机客户端系统主要功能需求如下：

(1) 区分已注册的用户和未注册的用户。用户注册并登录后，才能运行该系统的全部功能，未注册的用户仅能浏览部分页面。

(2) 区分已登录的用户和未登录的用户。已注册的用户未进行登录，仅能浏览部分页面，已注册的用户登录后，才能运行该系统的全部功能。

(3) 已登录的用户进行网上报名。

(4) 已登录的用户查询已录用、未录用、已报名等求职消息。

(5) 已登录的用户查询和编辑个人简历,注销登录。

(6) 已登录的用户和未登录的用户都可以浏览兼职和全职的职位概况、职位详情。

(7) 已登录的用户和未登录的用户都可以浏览"关于产品"页的内容。

10.1.2 总体设计

1. 设计目标

网上求职手机客户端系统,应达到方便用户使用、界面清晰美观、系统安全可靠、运行速度快效率高、节省存储空间等目标。

2. 技术特点

(1) 采用 Fragment 划分用户界面,以进行动态、灵活的用户界面设计。

(2) 采用 listView 和 Adapter,以展示多个公司的图标和职位概况。

(3) 采用散列映射和数组,以处理登录数据,用户数据、兼职数据、全职数据等。

(4) 采用 SQLite 数据库存储和查询数据。

(5) 采用 Intent 传递数据,以区分已登录状态和未登录状态。

3. 系统设计

在需求分析基础上进行系统设计,系统模块结构图如图 10.1 所示。

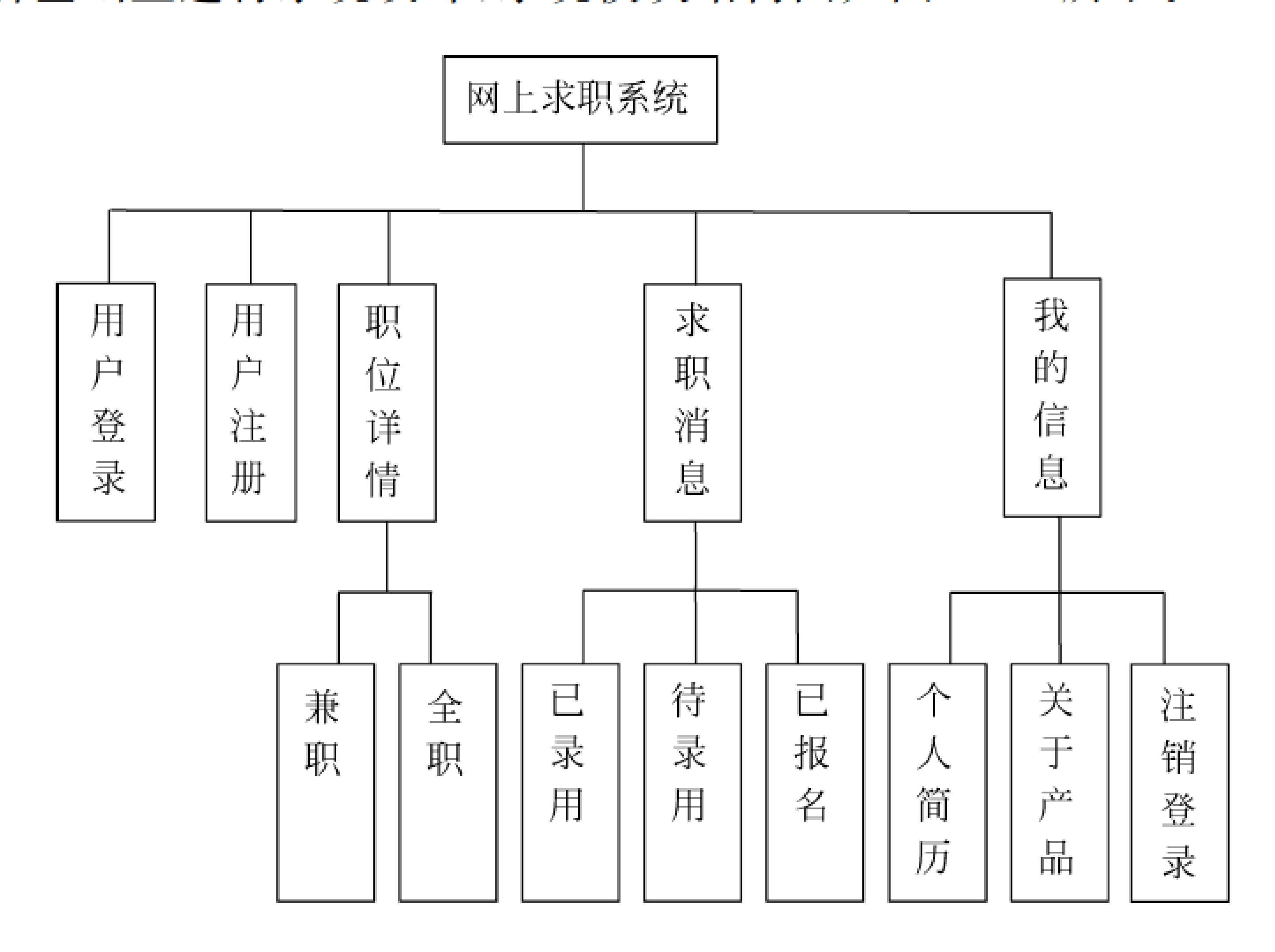


图 10.1 网上求职手机客户端系统总体结构图

10.1.3 数据库设计

1. 概念设计

此处略去网上求职手机客户端系统 E-R 图。

2. 逻辑设计

在逻辑设计中,网上求职手机客户端系统 E-R 图转换为以下关系模式:

user_table(_id, tel, pwd, name, sex, age, degrees, experience, phone)

3. 表结构设计和样本数据

上述关系模式对应的表结构如表 10.1 所示。

表 10.1 user_table

列 名	数据类型	是否主键	说 明
_id	integer	主键	自动递增
tel	varchar(20)		手机号码
pwd	varchar(20)		密码
name	varchar(20)		用户名
sex	varchar(20)		性别
age	varchar(20)		年龄
degrees	varchar(50)		学历
experience	varchar(50)		工作经历
phone	varchar(20)		电话号码

10.2 网上求职手机客户端系统程序结构设计

为开发网上求职手机客户端系统，在 Eclipse 中创建一个应用项目 OnlineCareer，包名为 com.application.onlinecareer，其项目目录树如图 10.2 所示。

网上求职手机客户端系统程序结构包含以下 4 类文件：Activity 类和 Fragment 类文件，Adapter 类和公共数据类文件，布局文件和其他资源文件。

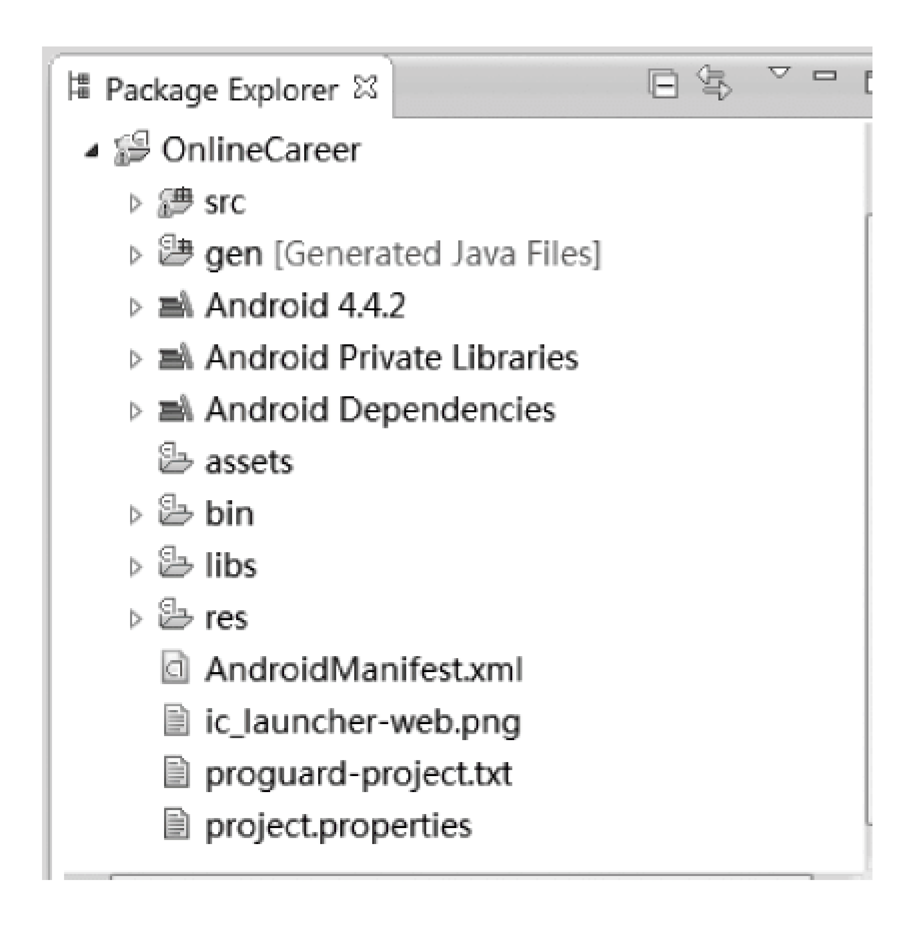


图 10.2 OnlineCareer 项目目录树

10.2.1 Activity 类和 Fragment 类

Activity 类和 Fragment 类共创建 10 个文件，如图 10.3 所示。

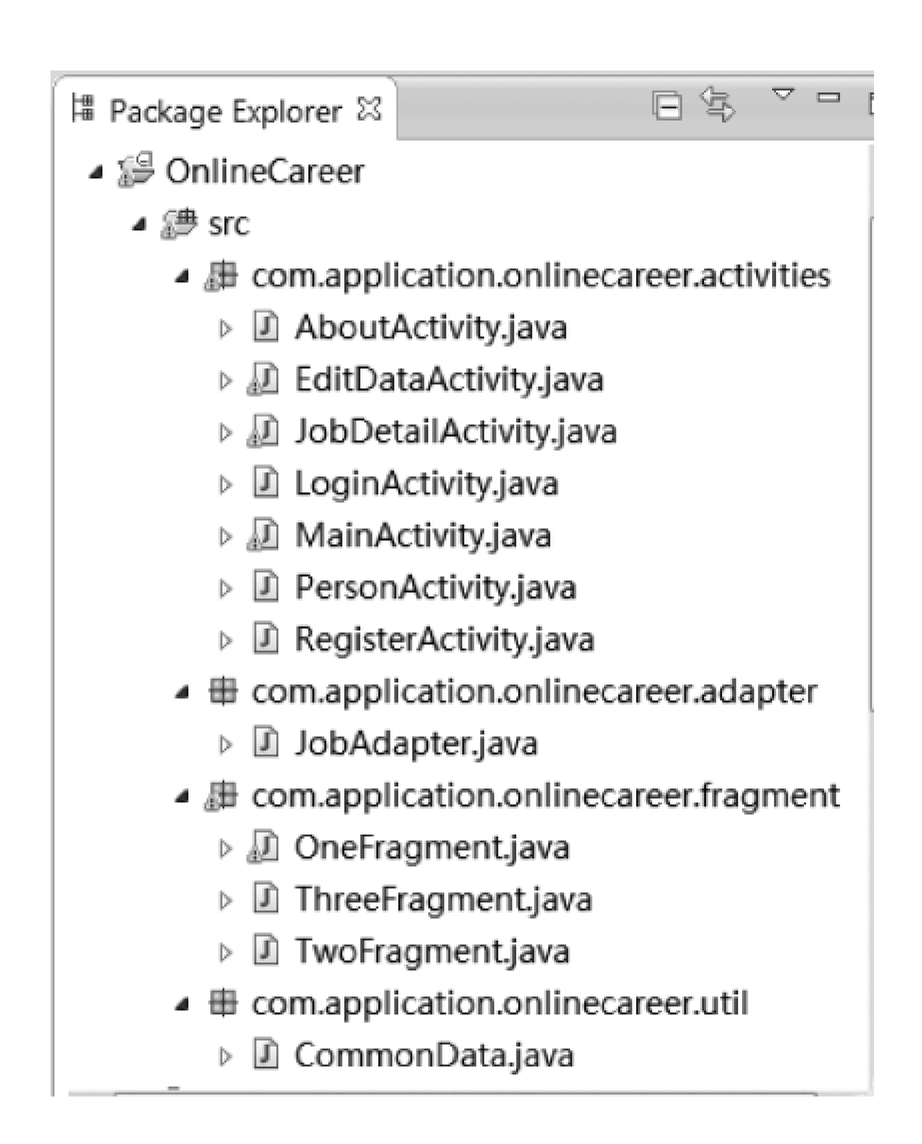


图 10.3　OnlineCareer 项目目录树-展开 src 目录

1. Activity 类

在 src 目录的 com. application. onlinecareer. activities 包下，创建 7 个 Activity 类文件。

(1) MainActivity. java：显示 activity_main 布局文件提供的用户界面，创建数据库和表，为其中的控件设置监听器。

(2) LoginActivity. java：显示 activity_login 登录布局文件提供的用户界面，为其中的控件设置监听器。

(3) RegisterActivity. java：显示 activity_register 注册布局文件提供的用户界面，为其中的控件设置监听器。

(4) JobDetailActivity. java：显示. activity_job_detail 职位详情布局文件提供的用户界面，为其中的控件设置监听器。

(5) PersonActivity. java：显示 activity_person 个人简历布局文件提供的用户界面，为其中的控件设置监听器。

(6) EditDataActivity. java：显示 activity_edit_data 编辑资料布局文件提供的用户界面，为其中的控件设置监听器。

(7) AboutActivity. java：显示 activity_about 关于产品布局文件提供的用户界面，为其中的控件设置监听器。

2. Fragment 类

在 src 目录的 com. application. onlinecareer. fragment 包下，创建三个 Fragment 类文件。

(1) OneFragment. java：显示 fragment_one 布局文件提供的用户界面，为其中的控件设置监听器。

(2) TwoFragment. java：显示 fragment_two 布局文件提供的用户界面，为其中的控件设置监听器。

(3) ThreeFragment.java：显示 fragment_three 布局文件提供的用户界面，为其中的控件设置监听器。

10.2.2 Adapter 类和公共数据类

Adapter 类和公共数据类分别创建一个文件，如图 10.3 所示。

1. Adapter 类

在 src 目录的 com.application.onlinecareer.adapter 包下，创建一个文件。

JobAdapter.java：定义 JobAdapter 类，其中定义方法：getCount()、getItem()、getItemId()、getView()，定义内部类 ViewHolder。

2. 公共数据类

在 src 目录的 com.application.onlinecareer.util 包下，创建一个文件。

CommonData.java：定义 CommonData 类，其中定义的公共数据对象有：登录状态 isLogin，登录用户数据 user_hashMap，所有用户数据 list，用户数据库 db，兼职数据 part_job_img、part_job_name、part_job_company、part_job_time、part_job_salary、part_job_people、part_job_address、part_job_payway，全职数据 full_job_img、full_job_name、full_job_company、full_job_time、full_job_salary、full_job_people、full_job_address、full_job_payway。

10.2.3 布局文件

布局文件是重要的资源文件，在 res/layout 目录下创建 12 个布局文件，如图 10.4 所示。

1. Activity 引用的布局文件

(1) activity_main.xml：定义 activity_main 布局文件，包含"首页"单选按钮、"消息"单选按钮、"我的"单选按钮等。

(2) activity_login.xml：定义 activity_main 布局文件，包含"登录"标题条，"手机号码"编辑框、"密码"编辑框，"登录"按钮，"注册"文本框等。

(3) activity_register.xml：定义 activity_register 布局文件，包含"注册"标题条，"手机号码"文本框和编辑框、"密码"文本框和编辑框、"确认密码"文本框和编辑框，"注册"按钮等。

(4) activity_job_detail.xml：定义 activity_job_detail 布局文件，包含"职位详情"标题条，"职位名称"文本框、"兼职/全职"文本框、"公司名称"文本框、"工资"文本框，"人数"文本框、"时间"文本框、"地址"文本框、"结算"文本框，"我要报名"文本框等。

(5) activity_person.xml：定义 activity_person 布局文件，包含"个人简历"标题条，"用户名"文本框、"性别"文本框、"年龄"文本框、"学历"文本框、"工作经历"文本框、"电话号码"文本框等。

(6) activity_edit_data.xml：定义 activity_edit_data 布局文件，包含"编辑资料"标题条，"用户名"文本框和编辑框、"性别"文本框和"男"单选按钮及"女"单选按钮、"年龄"文本框和编辑框、"学历"文本框和编辑框、"工作经历"文本框和编辑框、"电话号码"文本框和编

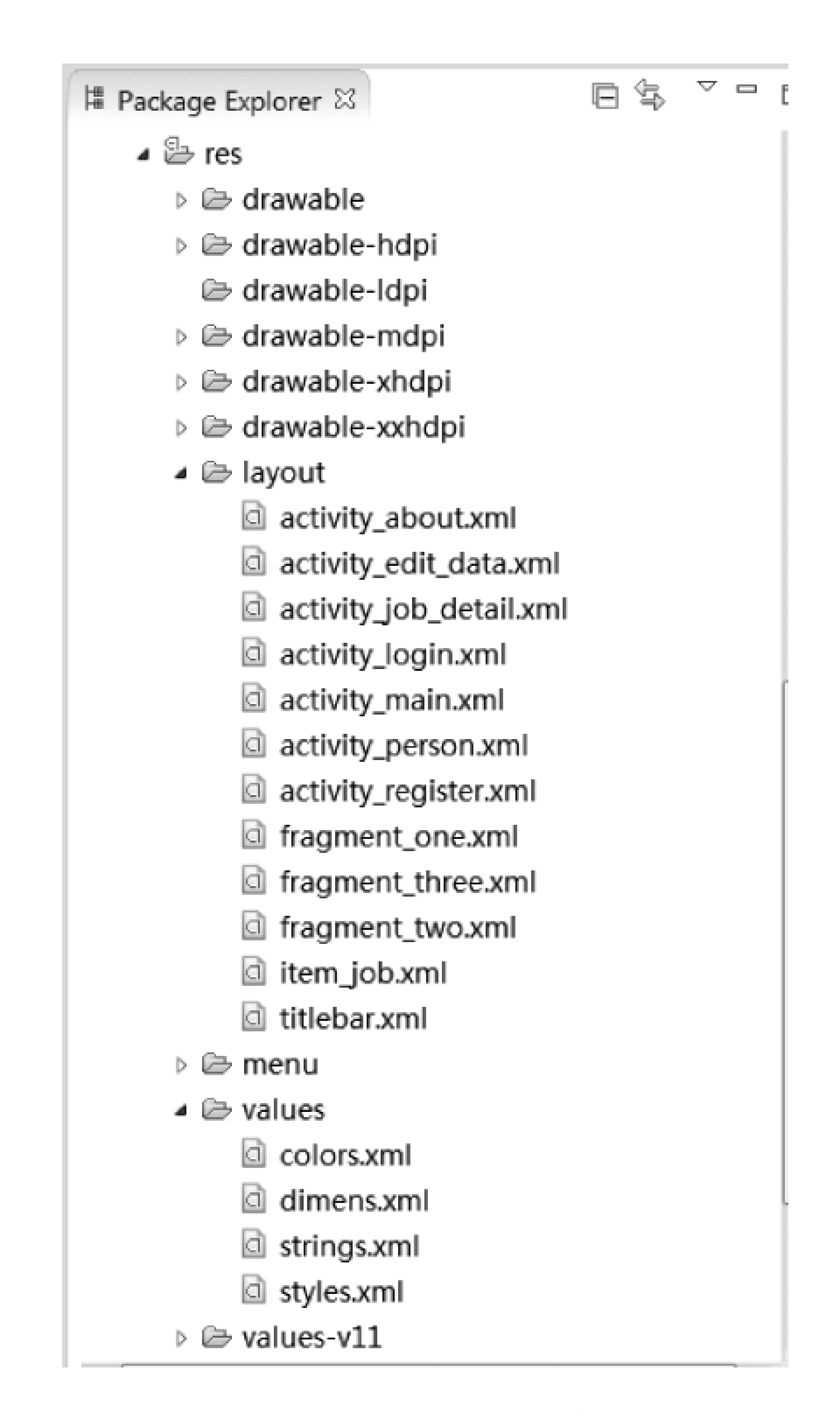


图 10.4　OnlineCareer 项目目录树-展开 res 目录

辑框等。

（7）activity_about.xml：定义 activity_about 布局文件，包含“关于产品”标题条，“关于产品”文本框等。

2. Fragment 引用的布局文件

（1）fragment_one.xml：定义 fragment_one 布局文件，包含“首页”文本框、“兼职”文本框、“全职”文本框和列表视图等。

（2）fragment_two.xml：定义 fragment_two 布局文件，包含“消息”文本框、“已录用”文本框、“待录用”文本框、“已报名”文本框和列表视图等。

（3）fragment_three.xml：定义 fragment_three 布局文件，包含“我的”文本框、“未登录”文本框、“我的简历”文本框、“关于产品”文本框、“撤销登录”文本框和列表视图等。

3. 其他布局文件

（1）item_job.xml：定义 item_job 布局文件，包含“职位名称”文本框、“兼职/全职”文本框、“公司名称”文本框，“时间”文本框、“工资”文本框等。

（2）titlebar.xml：定义 titlebar 布局文件，包含“返回”文本框、“标题”文本框、“进入”文本框等。

10.2.4　其他资源文件

资源文件除布局文件外，还有其他资源文件，介绍如下。

1. 在 res/values 目录下创建的资源文件

在该目录下创建 4 个资源文件，如图 10.4 所示。

(1) strings. xml：定义字符串。

(2) colors. xml：定义颜色。

(3) dimens. xml：定义尺寸。

(4) styles. xml：定义样式。

2. 在 res/ drawable-xxhdpi 目录下复制的图片文件

在该目录下复制 12 个图片文件。

10.3 基本页面

为了实现灵活、动态的界面设计，网上求职手机客户端系统设计采用 Fragment 划分用户界面，将 Activity 提供的用户界面划分为上下两个部分，在 Activity 运行时，Fragment 将嵌入到用户界面的上部。

基本页面是用户界面中的重要部分，基本页面由首页、消息页、我的页组成，下面介绍基本页面的开发。

10.3.1 首页

启动网上求职手机客户端系统，即进入首页，或当“首页”单选按钮被选中，也进入首页，此时，OneFragment 将嵌入到首页上部。

首页由首页上部和首页下部组成，首页上部包含“兼职”“全职”“职位名称”“公司名称”“时间”“工资”等栏目，首页下部为一组单选按钮：“首页”单选按钮、“消息”单选按钮、“我的”单选按钮。

首页开发步骤如下。

1. 设计布局

首页布局文件为 activity_main. xml 和 fragment_one. xml。

(1) 在 res/layout 目录下的 activity_main. xml 文件中，上部添加一个 FrameLayout 容器，下部添加一组单选按钮。在该文件中编辑代码如下：

```
<!-- 定义垂直分布的线性布局 -->
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:tools="http://schemas.android.com/tools"
 android:id="@+id/container"
 android:layout_width="match_parent"
 android:layout_height="match_parent"
 android:orientation="vertical"
 tools:context="com.example.test.MainActivity"
 tools:ignore="MergeRootFrame" >

        <!-- 设置帧布局,用作 Fragment 布局嵌入 -->
```

```
        <FrameLayout
            android:id="@+id/fl_content"
            android:layout_width="fill_parent"
            android:layout_height="0dp"
            android:layout_weight="1"></FrameLayout>
        <View
            android:layout_width="match_parent"
            android:layout_height="0.1dp"
            android:background="@color/basecolor"/>

        <!-- 定义单选按钮组 -->
        <RadioGroup
            android:background="@color/white"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="horizontal">
            <!-- 设置"首页"单选按钮 -->
            <RadioButton
                android:id="@+id/radioButton1"
                style="@style/main_tab"
                android:text="首页"
                android:drawableTop="@drawable/home_normal" />
            <!-- 设置"消息"单选按钮 -->
            <RadioButton
                android:id="@+id/radioButton2"
                style="@style/main_tab"
                android:text="消息"
                android:drawableTop="@drawable/message_normal" />
            <!-- 设置"我的"单选按钮 -->
            <RadioButton
                android:id="@+id/radioButton3"
                style="@style/main_tab"
                android:text="我的"
                android:drawableTop="@drawable/my_normal" />
        </RadioGroup>
    </LinearLayout>
```

(2) 在 res/layout 目录下的 fragment_one.xml 文件中,设置“首页”文本框、“兼职”文本框、“全职”文本框和列表视图。在该文件中编辑代码如下:

```
<!-- 定义垂直分布的线性布局 -->
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
        xmlns:tools="http://schemas.android.com/tools"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical"
```

```
        android:background = "@color/white"
        tools:context = "cn.com.mushroom.demo.fragment.OneFragment">

        <!-- 设置"首页"文本框 -->
        < TextView
            android:layout_width = "match_parent"
            android:layout_height = "50dp"
            android:background = "@color/basecolor"
            android:textColor = "@color/white"
            android:textSize = "18sp"
            android:gravity = "center"
            android:text = "首\u3000 页"/>

        < LinearLayout
            android:layout_width = "match_parent"
            android:layout_height = "50dp"
            android:orientation = "horizontal">

            <!-- 设置"兼职"文本框 -->
            < TextView
                android:id = "@ + id/tv_part_job"
                android:layout_width = "0dp"
                android:layout_height = "match_parent"
                android:layout_weight = "1"
                android:gravity = "center"
                android:textSize = "18sp"
                android:textColor = "@color/basecolor"
                android:text = "兼 职"/>

            <!-- 设置"全职"文本框 -->
            < TextView
                android:id = "@ + id/tv_full_time"
                android:layout_width = "0dp"
                android:layout_height = "match_parent"
                android:layout_weight = "1"
                android:textSize = "18sp"
                android:gravity = "center"
                android:text = "全 职"/>
        </LinearLayout >

        < View
            android:layout_width = "match_parent"
            android:layout_height = "0.3dp"
            android:background = "@color/basecolor" />

```

```
        <!-- 设置列表视图 -->
        <ListView
            android:id="@+id/lv_job"
            android:layout_width="match_parent"
            android:layout_height="match_parent"></ListView>
</LinearLayout>
```

2. 编辑代码

首页的 Activity 类文件为 MainActivity.java，Fragment 类文件为 OneFragment.java。

（1）在 com.application.onlinecareer.activities 包下的 MainActivity.java 文件中，加载 activity_main.xml 布局文件并实现事件监听器接口，当 Activity 运行时，将 OneFragment 添加到 Activity 布局的上部；下部为一组单选按钮对象设置监听器。在该文件中编辑代码如下：

```
package com.application.onlinecareer.activities;

import com.application.onlinecareer.R;
import com.application.onlinecareer.fragment.OneFragment;
import com.application.onlinecareer.fragment.ThreeFragment;
import com.application.onlinecareer.fragment.TwoFragment;
import com.application.onlinecareer.util.CommonData;
import android.support.v7.app.ActionBarActivity;
import android.view.KeyEvent;
import android.widget.CompoundButton;
import android.widget.CompoundButton.OnCheckedChangeListener;
import android.widget.FrameLayout;
import android.widget.RadioButton;
import android.annotation.SuppressLint;
import android.app.Fragment;
import android.app.FragmentManager;
import android.app.FragmentTransaction;
import android.content.Intent;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.os.Bundle;

//定义一个类 MainActivity 继承 ActionBarActivity 类，且实现事件监听器接口
@SuppressLint("NewApi")
public class MainActivity extends ActionBarActivity implementsOnCheckedChangeListener {
    private FrameLayout frameLayout;
    public OneFragment oneFragment;
    public TwoFragment twoFragment;
    public ThreeFragment threeFragment;
    private RadioButton radioButton1, radioButton2, radioButton3;
    private int currentTab = 0;              //设置选中单选按钮的标识
```

```
    //重写 onCreate()方法
    @Override
    protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
                                    //调用 setContentView()方法引用 activity_ main 布局

        //创建或打开数据库
        CommonData.db = SQLiteDatabase.openOrCreateDatabase(this.getFilesDir().toString()
                  + "/stu.db", null);
            createTable(CommonData.db);
            initView();
            initEvent();
        }
        //创建表
        private void createTable(SQLiteDatabase db) {
            //判断表是否存在,如果存在则不创建
            if (tableIsExist(db,"user_table"))
                return;
            //创建表 SQL 语句
            String stu_table = "create table user_table(_id integer primary key autoincrement," +
                "tel varchar(50)," + "pwd varchar(50)," + "name varchar(50)," + "sex varchar(50)," +
                "age varchar(50)," + "degrees varchar(50)," + "experience varchar(50)," +
                "phone varchar(50))";
            //执行 SQL 语句
            db.execSQL(stu_table);
        }
        //定义 tableIsExist()方法
        public boolean tableIsExist(SQLiteDatabase db,String tableName) {
            boolean result = false;
            if (tableName == null) {
                return false;
            }
            try {
                String sql = "select count( * ) as c from Sqlite_master where type = 'table' and
                        name = '" + tableName.trim() + "'";
                Cursor cursor = db.rawQuery(sql, null);
                if (cursor.moveToNext()) {
                    int count = cursor.getInt(0);
                    if (count > 0) {
                        result = true;
                    }
                }
                cursor.close();
```

```
            } catch (Exception e) {
              }
            return result;
        }

        //重写 onResume()方法
        @Override
        protected void onResume() {

            super.onResume();
            setSelect(currentTab);
        }

        //获取布局和控件的唯一标识
        private void initView() {
            frameLayout = (FrameLayout) findViewById(R.id.fl_content);
            radioButton1 = (RadioButton) findViewById(R.id.radioButton1);
            radioButton2 = (RadioButton) findViewById(R.id.radioButton2);
            radioButton3 = (RadioButton) findViewById(R.id.radioButton3);
        }

        private void initEvent() {
           //分别为单选按钮对象 radioButton1、radioButton2、radioButton3 设置监听器
           radioButton1.setOnCheckedChangeListener(this);
           radioButton2.setOnCheckedChangeListener(this);
           radioButton3.setOnCheckedChangeListener(this);
           setSelect(0);
           radioButton1.setCompoundDrawablesWithIntrinsicBounds(null, getApplication().
               getResources().getDrawable(R.drawable.home_selected), null, null);
           radioButton1.setTextColor(0xff2C97D4);
       }

       //设置图片和单选按钮的颜色
       private void setImg() {
           radioButton1.setCompoundDrawablesWithIntrinsicBounds(null, getApplication().
               getResources().getDrawable(R.drawable.home_normal), null, null);
           radioButton2.setCompoundDrawablesWithIntrinsicBounds(null, getApplication().
               getResources().getDrawable(R.drawable.message_normal), null, null);
           radioButton3.setCompoundDrawablesWithIntrinsicBounds(null, getApplication().
               getResources().getDrawable(R.drawable.my_normal), null, null)
           radioButton1.setTextColor(0xff979797);
           radioButton2.setTextColor(0xff979797);
           radioButton3.setTextColor(0xff979797);
       }

```

```
//重写 onCheckedChanged ()方法
//依据选中的单选按钮,确定 currentTab 的值,调用 setSelect()方法
@Override
public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
    if (isChecked) {
        switch (buttonView.getId()) {
        case R.id.radioButton1:
          setImg();
          currentTab = 0;
          radioButton1.setTextColor(0xff2C97D4);
          radioButton1.setCompoundDrawablesWithIntrinsicBounds(null,getApplication().
                getResources().getDrawable(R.drawable.home_selected), null, null);
          setSelect(0);
          break;
        case R.id.radioButton2:
          if (CommonData.isLogin) {
            setImg();
            currentTab = 1;
            radioButton2.setTextColor(0xff2C97D4);
            radioButton2.setCompoundDrawablesWithIntrinsicBounds(null,
                    getApplication().getResources().getDrawable(
                    R.drawable.message_selected), null, null);
            setSelect(1);
            } else {
                Intent intent = new Intent(this, LoginActivity.class);
                startActivityForResult(intent, 1);
            }
            break;
        case R.id.radioButton3:
          setImg();
          currentTab = 2;
          radioButton3.setTextColor(0xff2C97D4);
          radioButton3.setCompoundDrawablesWithIntrinsicBounds(null,getApplication().
                getResources().getDrawable(R.drawable.my_selected), null, null);
          setSelect(2);
          break;
        }
    }
}

//处理从 LoginActivity 返回的数据
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    String result = data.getExtras().getString("result");
    if (result.equals("ok")) {
```

```
            setImg();
            currentTab = 1;
            radioButton2.setTextColor(0xff2C97D4);
            radioButton2.setCompoundDrawablesWithIntrinsicBounds(null,getApplication().
                    getResources().getDrawable(R.drawable.message_selected),null,null);
            setSelect(1);
        } else {
            if (currentTab == 0) {
                radioButton1.setChecked(true);
                radioButton1.setTextColor(0xff2C97D4);
                radioButton1.setCompoundDrawablesWithIntrinsicBounds(null,getApplication().
                    getResources().getDrawable(R.drawable.home_selected), null, null);
                setSelect(0);
            } else if (currentTab == 2) {
                radioButton3.setChecked(true);
                radioButton3.setTextColor(0xff2C97D4);
                radioButton3.setCompoundDrawablesWithIntrinsicBounds(null, getApplication().
                    getResources().getDrawable(R.drawable.my_selected), null, null);
                setSelect(2);
            }
        }
    }

    //定义 setSelect()方法,依据调用该方法的实参值,确定添加的 Fragment
    public void setSelect(int i) {
        FragmentManager fm = getFragmentManager();
        FragmentTransaction transaction = fm.beginTransaction();
        switch (i) {
        case 0:
            hideFragment(transaction, twoFragment);
            hideFragment(transaction, threeFragment);
            if (oneFragment == null) {
                oneFragment = new OneFragment();
                transaction.add(R.id.fl_content, oneFragment, "oneFragment");
            } else {
                transaction.show(oneFragment);
            }
            break;
        case 1:
            hideFragment(transaction, oneFragment);
            hideFragment(transaction, threeFragment);
            if (twoFragment == null) {
                twoFragment = new TwoFragment();
                transaction.add(R.id.fl_content, twoFragment, "twoFragment");
            } else {
```

```
211                 transaction.show(twoFragment);
212             }
213             break;
214         case 2:
215             hideFragment(transaction, oneFragment);
216             hideFragment(transaction, twoFragment);
217             if (threeFragment == null) {
218                 threeFragment = new ThreeFragment();
219                 transaction.add(R.id.fl_content, threeFragment, "threeFragment");
220             } else {
221                 transaction.show(threeFragment);
222             }
223             break;
224         }
225         transaction.commit();
226     }
227
228     private void hideFragment(FragmentTransaction transaction, Fragment fragment) {
229         if (fragment != null) {
230             transaction.hide(fragment);
231         }
232     }
233
234      //单击退出键退出界面的时候不会关闭,应用下次进入的时候不用重启
235     @Override
236     public boolean dispatchKeyEvent(KeyEvent event) {
237         if (event.getKeyCode() == KeyEvent.KEYCODE_BACK
238                 && event.getAction() == KeyEvent.ACTION_DOWN
239                 && event.getRepeatCount() == 0) {
240             moveTaskToBack(true);
241             return true;
242         }
243         return super.dispatchKeyEvent(event);
244     }
245 }
```

① 第 24 行至第 245 行定义一个类 MainActivity 继承 ActionBarActivity 类，且实现事件监听器接口。

② 第 33 行至第 45 行重写 onCreate()方法，该方法在创建 Activity 时被回调，且只会被调用一次。其中，第 37 行调用 setContentView()方法引用 activity_main 布局，第 40 行至第 41 行创建或打开数据库，第 42 行调用 createTable()方法创建表，第 43 行调用 initView()方法，第 44 行调用 initEvent()方法。

③ 第 47 行至第 57 行定义 createTable()方法创建表，第 52 行至第 54 行为创建表 SQL 语句，第 56 行为执行 SQL 语句。

④ 第 82 行至第 87 行重写 onResume()方法，该方法当 Activity 可以开始与用户进行交互之前被调用，第 86 行调用 setSelect()方法。

⑤ 第 90 行至第 95 行定义 initView()方法，获取 FrameLayout 布局的唯一标识，并分别获取三个 RadioButton——首页”单选按钮、“消息”单选按钮、“我的”单选按钮的唯一标识。

⑥ 第 97 行至第 106 行定义 initEvent()方法，分别为单选按钮对象 radioButton1、radioButton2、radioButton3 设置监听器。

⑦ 第 123 行至第 159 行重写 onCheckedChanged()方法，依据选中的单选按钮，确定 currentTab 的值，调用 setSelect()方法。

⑧ 第 163 行至第 187 行重写 onActivityResult()方法，处理用户登录成功后，从登录页通过 Intent 返回的数据。

⑨ 第 190 行至第 226 行定义 setSelect()方法，依据调用该方法的实参值，确定添加或显示的 Fragment。第 194 行至第 203 行，当 i 为 0 时，在 Activity 中添加或显示 OneFragment。第 204 行至第 213 行，当 i 为 1 时，在 Activity 中添加或显示 TwoFragment。第 214 行至第 224 行，当 i 为 2 时，在 Activity 中添加或显示 ThreeFragment。

(2) 下面的 OneFragment 将会加载 fragment_one.xml 布局文件，构成 Activity 界面的上部，分别为兼职列表对象和全职列表对象赋值，创建适配器对象 adapter 绑定兼职列表和全职列表数据，将适配器对象 adapter 添加到列表视图对象并显示出来。

在 com.application.onlinecareer.fragment 包下的 OneFragment.java 文件中编辑代码如下：

```
package com.application.onlinecareer.fragment;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import com.application.onlinecareer.activities.JobDetailActivity;
import com.application.onlinecareer.adapter.JobAdapter;
import com.application.onlinecareer.util.CommonData;
import com.application.onlinecareer.R;
import android.annotation.SuppressLint;
import android.app.Fragment;
import android.content.Intent;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.View.OnClickListener;
import android.view.ViewGroup;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.ListView;
import android.widget.TextView;
```

```

//定义一个类 OneFragment 继承 Fragment 类,且实现事件监听器接口
@SuppressLint("NewApi")
public class OneFragment extends Fragment implements OnClickListener{
    private View rootView;
    private TextView tv_part_job,tv_full_time;
    private ListView lv_job;                    //定义列表视图对象 lv_job
    private JobAdapter adapter;                 //定义适配器对象 adapter
    private List<HashMap<String, Object>> part_job_Data;
                                                        //定义兼职列表对象 part_job_Data
    private List<HashMap<String, Object>> full_job_Data;
                                                        //定义全职列表对象 full_job_Data
    private int job_type = 0;                   //0 代表兼职,1 代表全职

    //创建和返回与 Fragment 关联的 View 对象
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
            Bundle savedInstanceState) {
        if (rootView == null) {
            rootView = inflater.inflate(R.layout.fragment_one, container, false);
        }
        return rootView;
    }

    //告诉 Fragment 对象,它所依附的 Activity 对象已经完成了 Activity.onCreate()
    //方法的执行
    public void onActivityCreated(Bundle savedInstanceState) {
        super.onActivityCreated(savedInstanceState);
        initData();
        initView();
        //为列表视图对象 lv_job 绑定监听器
        lv_job.setOnItemClickListener(new OnItemClickListener() {

            //重写 onItemClick()方法
            @Override
            public void onItemClick(AdapterView<?> adapter, View arg1, int position,
                    long arg3) {
                //通过 Intent 组件调用 JobDetailActivity 类,进入"职位详情"页
                Intent intent = new Intent(getActivity(),JobDetailActivity.class);
                Bundle data = new Bundle();
                data.putSerializable("job", (HashMap<String, Object>)
                        adapter.getItemAtPosition(position));
                adapter.getItemAtPosition(position));
                intent.putExtras(data);
                startActivity(intent);
            }
```

```
            });
        }
        //定义 initData()方法
        private void initData(){
                //创建兼职列表对象 part_job_Data
                part_job_Data = new ArrayList<HashMap<String, Object>>();
                //创建全职列表对象 full_job_Data
                full_job_Data = new ArrayList<HashMap<String, Object>>();

                //为兼职列表对象 part_job_Data 赋值
                for (int i = 0; i < CommonData.part_job_name.length; i++) {
                    HashMap<String, Object> hashMap = new HashMap<String, Object>();
                    hashMap.put("img",CommonData.part_job_img[i]);
                    hashMap.put("name",CommonData.part_job_name[i]);
                    hashMap.put("jobType","兼职");
                    hashMap.put("company",CommonData.part_job_company[i]);
                    hashMap.put("time",CommonData.part_job_time[i]);
                    hashMap.put("salary",CommonData.part_job_salary[i]);
                    hashMap.put("people",CommonData.part_job_people[i]);
                    hashMap.put("address",CommonData.part_job_address[i]);
                    hashMap.put("payWay",CommonData.part_job_payway[i]);
                    part_job_Data.add(hashMap);
                }

                //为全职列表对象 full_job_Data 赋值
                for (int i = 0; i < CommonData.full_job_name.length; i++) {
                    HashMap<String, Object> hashMap = new HashMap<String, Object>();
                    hashMap.put("img",CommonData.full_job_img[i]);
                    hashMap.put("name",CommonData.full_job_name[i]);
                    hashMap.put("jobType","全职");
                    hashMap.put("company",CommonData.full_job_company[i]);
                    hashMap.put("time",CommonData.full_job_time[i]);
                    hashMap.put("salary",CommonData.full_job_salary[i]);
                    hashMap.put("people",CommonData.full_job_people[i]);
                    hashMap.put("address",CommonData.full_job_address[i]);
                    hashMap.put("payWay",CommonData.full_job_payway[i]);
                   full_job_Data.add(hashMap);
               }
           }
        //定义 initView()方法
        private void initView() {
             tv_part_job = (TextView) rootView.findViewById(R.id.tv_part_job);
             tv_full_time = (TextView) rootView.findViewById(R.id.tv_full_time);
             lv_job = (ListView) rootView.findViewById(R.id.lv_job);
             tv_part_job.setOnClickListener(this);
```

```
109                 tv_full_time.setOnClickListener(this);
110 
111                 //创建适配器对象 adapter,并绑定兼职列表数据
112                 adapter = new JobAdapter(getActivity(), part_job_Data);
113             //将适配器对象 adapter 添加到列表视图对象 lv_job,并显示出来
114                 lv_job.setAdapter(adapter);
115         }
116 
117         //重写 onClick()方法
118         @Override
119         public void onClick(View v) {
120             setTextColor();
121             switch (v.getId()) {
122             case R.id.tv_part_job:
123                 tv_part_job.setTextColor(getResources().getColor(R.color.basecolor));
124                 job_type = 0;
125                 //创建适配器对象 adapter,并绑定兼职列表数据
126                 adapter = new JobAdapter(getActivity(), part_job_Data);
127                 //将适配器对象 adapter 添加到列表视图对象 lv_job,并显示出来
128                 lv_job.setAdapter(adapter);
129                 break;
130             case R.id.tv_full_time:
131                 tv_full_time.setTextColor(getResources().getColor(R.color.basecolor));
132                 job_type = 1;
133                 //创建适配器对象 adapter,并绑定全职列表数据
134                 adapter = new JobAdapter(getActivity(), full_job_Data);
135                 //将适配器对象 adapter 添加到列表视图对象 lv_job,并显示出来
136                 lv_job.setAdapter(adapter);
137                 break;
138             }
139         }
140         //改变字体的背景颜色
141         public void setTextColor() {
142             tv_part_job.setTextColor(getResources().getColor(R.color.text_color_primary));
143             tv_full_time.setTextColor(getResources().getColor(R.color.text_color_primary));
144         }
145     }
```

① 第 23 行至第 145 行定义一个类 OneFragment 继承 Fragment 类,且实现事件监听器接口。

② 第 35 行至第 41 行定义 onCreateView()方法,创建和返回与 Fragment 关联的 View 对象 rootView。

③ 第 44 行至第 65 行定义 onActivityCreated 方法,该方法告诉 Fragment 对象,它所依附的 Activity 对象已经完成了 Activity.onCreate()方法的执行,第 46 行调用 initData()方

法，第 47 行调用 initView()方法。

④ 第 49 行至第 64 行为列表视图对象 lv_job 绑定监听器，第 52 行至第 63 行重写 onItemClick()方法，当单击某一项目时，通过 Intent 组件调用 JobDetailActivity 类，进入"职位详情"页。

⑤ 第 67 行至第 102 行定义 initData()方法，第 69 行创建兼职列表对象 part_job_Data，第 71 行创建全职列表对象 full_job_Data，第 74 行至第 86 行为兼职列表对象 part_job_Data 赋值，第 89 行至第 101 行为全职列表对象 full_job_Data 赋值。

⑥ 第 104 行至第 115 行定义 initView()方法，第 112 行创建适配器对象 adapter，并绑定兼职列表数据，第 114 行将适配器对象 adapter 添加到列表视图对象 lv_job，并显示出来。

⑦ 第 118 行至第 139 行重写 onClick()方法，第 122 行至第 130 行为单击"兼职"文本框时，创建适配器对象 adapter，并绑定兼职列表数据，将适配器对象 adapter 添加到列表视图对象 lv_job，并显示出来；第 130 行至第 137 行为单击"全职"文本框时，创建适配器对象 adapter，并绑定全职列表数据，将适配器对象 adapter 添加到列表视图对象lv_job，并显示出来。

3. 运行结果

在 Eclipse 中启动模拟器，然后运行项目 OnlineCareer，当"首页"单选按钮被选中，出现网上求职系统首页，如图 10.5 所示。



图 10.5 首页

10.3.2 消息页

启动网上求职手机客户端系统，若用户已登录，当"消息"单选按钮被选中，进入消息页，此时 TwoFragment 将嵌入到消息页上部；若用户未登录，当"消息"单选按钮被选中，进入

用户登录页。

消息页由消息页上部和消息页下部组成，其中，消息页上部包含“消息”“已录用”“待录用”“已报名”等栏目，消息页下部为一组单选按钮：“首页”单选按钮、“消息”单选按钮、“我的”单选按钮。

消息页开发步骤如下。

1. 设计布局

在消息页布局文件中，activity_main. xml 与首页同，参见首页部分，另外，此处略去其 fragment_two. xml。

2. 编辑代码

消息页的 Activity 类文件 MainActivity. java 与首页同，参见首页部分，另外，此处略去其 Fragment 类文件 TwoFragment. java。

3. 运行结果

在 Eclipse 中启动模拟器，然后运行项目 OnlineCareer，当“消息”单选按钮被选中，且已登录，出现网上求职系统消息页，如图 10.6 所示。



图 10.6　消息页

10.3.3　我的页

启动网上求职手机客户端系统，当“我的”单选按钮被选中，进入我的页，此时 ThreeFragment 将嵌入到我的页上部。

我的页由我的页上部和我的页下部组成，其中，我的页上部由“我的”“我的简历”“关于

产品”“撤销登录”等栏目组成，我的页下部为一组单选按钮：“首页”单选按钮、“消息”单选按钮、“我的”单选按钮。

“我的页”开发步骤如下。

1. 设计布局

我的页布局文件为 activity_main.xml 和 fragment_three.xml。

(1) 我的页布局文件 activity_main.xml 和首页相同，参见首页部分。

(2) 在 res/layout 目录下的 fragment_three.xml 文件中，设置“我的”文本框、“未登录”文本框、“个人简历”文本框、设置“关于产品”文本框和“注销登录”文本框。在该文件中编辑代码如下：

```
<!-- 定义垂直分布的线性布局 -->
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:tools="http://schemas.android.com/tools"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:orientation="vertical"
  android:background="@color/white"
  tools:context="cn.com.mushroom.demo.fragment.OneFragment" >

      <!-- 设置"我的"文本框 -->
      <TextView
          android:layout_width="match_parent"
          android:layout_height="50dp"
          android:background="@color/basecolor"
          android:textColor="@color/white"
          android:textSize="18sp"
          android:gravity="center"
          android:text="我\u3000的"/>

      <ImageView
          android:id="@+id/iv_head"
          android:layout_width="70dp"
          android:layout_height="70dp"
          android:layout_gravity="center_horizontal"
          android:layout_marginTop="15dp"
          android:background="@drawable/header" />

      <!-- 设置"未登录"文本框 -->
      <TextView
          android:id="@+id/tv_nickName"
          android:layout_width="wrap_content"
          android:layout_height="wrap_content"
          android:layout_gravity="center_horizontal"
          android:layout_marginTop="10dp"
```

```
        android:text = "未登录"
        android:textColor = "@color/text_color_primary" />

    < View
        android:layout_marginTop = "30dp"
        android:layout_width = "match_parent"
        android:layout_height = "1dp"
        android:background = "@color/divider" />

    <!-- 设置"个人简历"文本框 -->
    < TextView
        android:id = "@ + id/tv_person"
        android:layout_width = "match_parent"
        android:layout_height = "45dp"
        android:background = "#ffffff"
        android:drawableEnd = "@drawable/ic_go_into"
        android:drawableLeft = "@drawable/ic_setting_information"
        android:drawablePadding = "5dp"
        android:gravity = "center_vertical"
        android:paddingLeft = "10dp"
        android:paddingRight = "10dp"
        android:singleLine = "true"
        android:text = "个人简历"
        android:textColor = "@color/text_color_primary"
        android:textSize = "@dimen/text_size_primary" />

    < View
        android:layout_width = "match_parent"
        android:layout_height = "1dp"
        android:background = "@color/divider" />

    <!-- 设置"关于产品"文本框 -->
    < TextView
        android:id = "@ + id/tv_about"
        android:layout_width = "match_parent"
        android:layout_height = "45dp"
        android:background = "#ffffff"
        android:drawableEnd = "@drawable/ic_go_into"
        android:drawableLeft = "@drawable/ic_release_category_requirements"
        android:drawablePadding = "5dp"
        android:gravity = "center_vertical"
        android:paddingLeft = "10dp"
        android:paddingRight = "10dp"
        android:singleLine = "true"
        android:text = "关于产品"
```

```
            android:textColor = "@color/text_color_primary"
            android:textSize = "@dimen/text_size_primary" />

        < View
            android:layout_width = "match_parent"
            android:layout_height = "1dp"
            android:background = "@color/divider" />

        <! -- 设置"注销登录"文本框 -->
        < TextView
            android:id = "@ + id/tv_loginout"
            android:layout_width = "match_parent"
            android:layout_height = "45dp"
            android:background = "#ffffff"
            android:drawableEnd = "@drawable/ic_go_into"
            android:drawableLeft = "@drawable/ic_setting_sign_out"
            android:drawablePadding = "5dp"
            android:gravity = "center_vertical"
            android:paddingLeft = "10dp"
            android:paddingRight = "10dp"
            android:singleLine = "true"
            android:text = "注销登录"
            android:textColor = "@color/text_color_primary"
            android:textSize = "@dimen/text_size_primary" />

        < View
            android:layout_width = "match_parent"
            android:layout_height = "1dp"
            android:background = "@color/divider" />
</LinearLayout >
```

2. 编辑代码

我的页的 Activity 类文件为 MainActivity. java，Fragment 类文件为 ThreeFragment. java。

(1) 我的页在 com. application. onlinecareer. activities 包下的 MainActivity. java 文件和首页相同，参见首页部分。

(2) 下面的 ThreeFragment 将会加载 fragment_three. xml 布局文件，构成 Activity 界面的上部。

在 com. application. onlinecareer. fragment 包下的 ThreeFragment. java 文件中编辑代码如下：

```
package com.application.onlinecareer.fragment;

import com.application.onlinecareer.activities.AboutActivity;
import com.application.onlinecareer.activities.PersonActivity;
import com.application.onlinecareer.util.CommonData;
```

```
import com.application.onlinecareer.R;
import android.annotation.SuppressLint;
import android.app.Fragment;
import android.content.Intent;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.View.OnClickListener;
import android.view.ViewGroup;
import android.widget.ImageView;
import android.widget.TextView;
import android.widget.Toast;

    //定义一个类 ThreeFragment 继承 Fragment 类,且实现事件监听器接口
    @SuppressLint("NewApi")
    public class ThreeFragment extends Fragment implements OnClickListener {
        private View rootView;
        private TextView tv_person, tv_about, tv_loginout, tv_nickName;
        private ImageView iv_head;

    //创建和返回与 Fragment 关联的 View 对象
        public View onCreateView(LayoutInflater inflater, ViewGroup container,
                Bundle savedInstanceState) {
            if (rootView == null) {
                rootView = inflater.inflate(R.layout.fragment_three, container, false);
            }
            return rootView;
        }

        //告诉 Fragment 对象,它所依附的 Activity 对象已经完成了 Activity.onCreate()
        //方法的执行
        public void onActivityCreated(Bundle savedInstanceState) {
            super.onActivityCreated(savedInstanceState);
            initView();
        }

        //该方法被回调后,Fragment 对象可与用户交互
        @Override
        public void onResume() {
            super.onResume();
            if (CommonData.isLogin)
                tv_nickName.setText(CommonData.user_hashMap.get("name").toString());
        }
        //定义 initView()方法
        private void initView() {
```

```
            tv_person = (TextView) rootView.findViewById(R.id.tv_person);
            tv_about = (TextView) rootView.findViewById(R.id.tv_about);
            tv_loginout = (TextView) rootView.findViewById(R.id.tv_loginout);
            tv_nickName = (TextView) rootView.findViewById(R.id.tv_nickName);
            iv_head = (ImageView)rootView.findViewById(R.id.iv_head);
            tv_person.setOnClickListener(this);
            tv_about.setOnClickListener(this);
            tv_loginout.setOnClickListener(this);
            iv_head.setOnClickListener(this);
        }

        //重写 onClick()方法
        @Override
        public void onClick(View v) {
            switch (v.getId()) {
            //单击匿名头像,如果已登录,通过 Intent 组件调用 PersonActivity 类,进入
            //"个人简历"页; 否则,显示"用户还未登录!"
            case R.id.iv_head:
                if(CommonData.isLogin){
                    Intent intent = new Intent(getActivity(), PersonActivity.class);
                    startActivity(intent);
                }else{
                    Toast.makeText(getActivity(), "用户还未登录!",
                            Toast.LENGTH_SHORT).show();
                }
                break;
            //单击"个人简历"文本框,如果已登录,通过 Intent 组件调用 PersonActivity
            //类,进入"个人简历"页; 否则,显示"用户还未登录!"
            case R.id.tv_person:
                if(CommonData.isLogin){
                    Intent intent = new Intent(getActivity(), PersonActivity.class);
                    startActivity(intent);
                }else{
                    Toast.makeText(getActivity(), "用户还未登录!",
                            Toast.LENGTH_SHORT).show();
                }
                break;
            //单击"关于产品"文本框,通过 Intent 组件调用 AboutActivity 类,进入"关于产品"页
            case R.id.tv_about:
                Intent intent = new Intent(getActivity(), AboutActivity.class);
                startActivity(intent);
                break;
            //单击"注销登录"文本框,置 isLogin 对象为 False,显示"退出登录!"
            case R.id.tv_loginout:
                CommonData.isLogin = false;
```

```
95                      CommonData.user_hashMap = null;
96                      tv_nickName.setText("未登录");
97                      Toast.makeText(getActivity(), "退出登录!", Toast.LENGTH_SHORT).show();
98                      break;
99                  }
100             }
101     }
```

① 第 20 行至第 101 行定义一个类 ThreeFragment 继承 Fragment 类，且实现事件监听器接口。

② 第 27 行至第 33 行定义 onCreateView()方法，创建和返回与 Fragment 关联的 View 对象 rootView。

③ 第 36 行至第 39 行定义 onActivityCreated 方法，该方法告诉 Fragment 对象，它所依附的 Activity 对象已经完成了 Activity.onCreate()方法的执行，第 38 行调用 initView()方法。

④ 第 42 行至第 47 行定义 onResume()方法，该方法被回调后，Fragment 对象可与用户交互，第 45 行至第 46 行当用户未登录时，该页上部文本框显示"未登录"。

⑤ 第 49 行至第 59 行定义 initView()方法，第 50 行至第 54 行分别通过 findViewById()方法获取控件的唯一标识，第 55 行至第 58 行分别为控件对象设置监听器。

⑥ 第 61 行至第 100 行重写 onClick()方法，第 67 行至第 75 行为单击匿名头像，如果已登录，通过 Intent 组件调用 PersonActivity 类进入"个人简历"页，否则，显示"用户还未登录!"；第 78 行至第 86 行为单击"个人简历"文本框，如果已登录，通过 Intent 组件调用 PersonActivity 类进入"个人简历"页，否则，显示"用户还未登录!"；第 88 行至第 91 行为单击"关于产品"文本框，通过 Intent 组件调用 AboutActivity 类，进入"关于产品"页；第 93 行至第 98 行为单击"注销登录"文本框，置 isLogin 对象为 False，显示"退出登录!"。

3. 运行结果

在 Eclipse 中启动模拟器，然后运行项目 OnlineCareer，当"我的"单选按钮被选中，出现网上求职系统"我的页"，如图 10.7 所示。



图 10.7　我的页

10.4　用户登录和注册

本节介绍用户登录和用户注册的内容。

10.4.1 用户登录

用户登录为已注册的用户使用本系统提供登录窗口。

启动网上求职手机客户端系统后，如果用户尚未登录，当“消息”单选按钮被选中，即进入用户登录页。

用户登录页由“返回”文本框、“手机号”编辑框、“密码”编辑框、“登录”按钮和“注册”文本框组成，其组成文件为 LoginActivity. java 和 activity_login. xml。

用户登录页开发步骤如下。

1. 设计布局

在 res/layout 目录下的 activity_login. xml 文件中，导入 titlebar 布局文件到本布局中，用于设置“返回”文本框、“登录”文本框。设置“手机号码”图标和编辑框，“请输入密码”图标和编辑框，“登录”按钮，“注册”文本框。

2. 编辑代码

在 com. application. onlinecareer. activities 包下的 LoginActivity. java 文件中，加载 activity_login. xml 布局文件，分别为控件对象绑定监听器，对输入数据进行校验和处理，验证用户是否已存在。在该文件中编辑代码如下：

```
package com.application.onlinecareer.activities;

import java.util.HashMap;
import com.application.onlinecareer.R;
import com.application.onlinecareer.util.CommonData;
import android.support.v7.app.ActionBarActivity;
import android.text.TextUtils;
import android.content.Intent;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;

    //定义一个类 LoginActivity 继承 ActionBarActivity 类，且实现事件监听器接口
    public class LoginActivity extends ActionBarActivity implements OnClickListener {

        private TextView tv_back, tv_title, tv_register;
        private EditText et_account, et_password;
        private Button btn_login;

```

```
    //重写 onCreate()方法
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        //调用 setContentView()方法引用 activity_login 布局
        setContentView(R.layout.activity_login);
        initView();
    }

    //定义 initView()方法
    private void initView() {

        //通过 findViewById()方法获取控件的唯一标识
        tv_title = (TextView) findViewById(R.id.tv_title);
        tv_title.setText(R.string.login);
        tv_back = (TextView) findViewById(R.id.tv_back);
        tv_register = (TextView) findViewById(R.id.tv_register);
        btn_login = (Button) findViewById(R.id.btn_login);

        tv_back.setOnClickListener(this);
        tv_register.setOnClickListener(this);
        btn_login.setOnClickListener(this);

        et_account = (EditText) findViewById(R.id.et_account);
        et_password = (EditText) findViewById(R.id.et_password);
    }

    //重写 onClick()方法
    @Override
    public void onClick(View v) {
        switch (v.getId()) {
        //使用 Intent 回传数据
        case R.id.tv_back:
            Intent intent = new Intent();
            //向 Intent 中添加一组键 - 值对附加信息
            intent.putExtra("result", "no");
            //打包回传数据,回传返回码 RESULT_OK 和 intent 对象
            setResult(RESULT_OK, intent);
            finish();
            break;
        case R.id.tv_register:
            intent = new Intent(this,RegisterActivity.class);
            startActivity(intent);
            break;
        case R.id.btn_login:
```

```
//使用 getText().toString()方法获取字符串信息
String tel = et_account.getText().toString();
String pwd = et_password.getText().toString();

if (TextUtils.isEmpty(tel)) {
    Toast.makeText(this, "手机号码不能为空!",
            Toast.LENGTH_SHORT).show();
    return;
}

if (TextUtils.isEmpty(pwd)) {
    Toast.makeText(this, "密码不能为空!", Toast.LENGTH_SHORT).show();
    return;
}
//定义服务器的电话号码、密码等属性
String service_tel = null;
String service_pwd = null;
String service_name = null;
String service_sex = null;
String service_age = null;
String service_school = null;
String service_major = null;
String service_phone = null;
HashMap<String, Object> hashMap = new HashMap<String, Object>();

if(isExist(CommonData.db,tel)){
     String sql = "select * from user_table where tel = '" + tel + "'";
   Cursor cursor = CommonData.db.rawQuery(sql, new String[0]);
   while (cursor.moveToNext()) {
     service_tel = cursor.getString(cursor.getColumnIndex("tel"));
       service_pwd = cursor.getString(cursor.getColumnIndex("pwd"));
       service_name = cursor.getString(cursor.getColumnIndex("name"));
       service_sex = cursor.getString(cursor.getColumnIndex("sex"));
       service_age = cursor.getString(cursor.getColumnIndex("age"));
       service_school = cursor.getString(cursor.getColumnIndex("school"));
       service_major = cursor.getString(cursor.getColumnIndex("major"));
       service_phone = cursor.getString(cursor.getColumnIndex("phone"));

       hashMap.put("tel", service_tel);
       hashMap.put("pwd", service_pwd);
       hashMap.put("name", service_name);
       hashMap.put("sex", service_sex);
       hashMap.put("age", service_age);
       hashMap.put("school", service_school);
       hashMap.put("major", service_major);
```

```
116                     hashMap.put("phone", service_phone);
117                 }
118                 cursor.close();
119             }else{
120                 return;
121             }
122 
123             //判断用户输入的手机号码、密码和服务器保存的手机号码、密码是否相等
124             if(tel.equals(service_tel) && pwd.equals(service_pwd)){
125                 CommonData.isLogin = true;
126                 Toast.makeText(this, "登录成功!", Toast.LENGTH_SHORT).show();
127 
128                 //存储用户登录的信息
129                 CommonData.user_hashMap = hashMap;
130 
131                 //使用 Intent 回传数据
132                 intent = new Intent();
133                 //向 Intent 中添加一组键-值对附加信息
134                 intent.putExtra("result", "ok");
135                 //打包回传数据,回传返回码 RESULT_OK 和 intent 对象
136                 setResult(RESULT_OK, intent);
137                 finish();
138             }else{
139                 Toast.makeText(this, "账号/密码错误!", Toast.LENGTH_SHORT).show();
140             }
141             break;
142         }
143     }
144 
145     //验证用户是否已存在
146     private boolean isExist(SQLiteDatabase db,String tel) {
147         String sql = "select * from user_table where tel = '" + tel + "'";
148         Cursor cursor = db.rawQuery(sql, new String[0]);
149         if (cursor.getCount() != 0) {
150             cursor.close();
151             return true;
152         }else{
153             Toast.makeText(LoginActivity.this, "用户不存在", Toast.LENGTH_SHORT).show();
154             cursor.close();
155             return false;
156         }
157     }
158 }
```

① 第 20 行至第 158 行定义一个类 LoginActivity 继承 ActionBarActivity 类,且实现事

件监听器接口。

② 第 27 行至第 33 行重写 onCreate()方法,其中,第 29 行调用 setContentView()方法引用 activity_login 布局,第 32 行调用 initView()方法。

③ 第 36 行至第 51 行定义 initView()方法,其中,第 38 行至第 43 行和第 49 行至第 50 行分别通过 findViewById()方法获取控件的唯一标识,第 45 行至第 47 行分别为控件对象设置监听器。

④ 第 54 行至第 143 行重写 onClick()方法。

⑤ 第 57 行至第 65 行当单击"返回"文本框时,使用 Intent 回传数据,第 61 行向 Intent 中添加一组键-值对附加信息:键名为 result、键值为 no,第 63 行打包回传数据,回传返回码 RESULT_OK 和 intent 对象,第 64 行将打包好的数据回传给 MainActivity,并运行其中的 onActivityResult()里的代码。

⑥ 第 66 行至第 69 行当单击"注册"文本框时,使用 Intent 显式启动 RegisterActivity,即启动"注册"页。

⑦ 第 70 行至第 141 行当单击"登录"文本框时,第 72 行至第 73 行使用 getText().toString()方法分别为字符串对象 tel 和 pwd 获取用户输入的手机号码和密码信息;第 75 行至第 84 行分别判断字符串对象 tel 和 pwd 是否为空,如果为空,则分别提示"手机号码不能为空!""密码不能为空!";第 96 行至第 121 行在用户数据库 CommonData.db 中查询 tel 值,返回的游标指针移动到数据库中用户输入的手机号码所在行,分别根据各个属性名称返回各个属性值,分别向散列映射对象 hashMap 中添加键-值对;第 124 行至第 140 行判断用户输入的手机号码、密码和服务器保存的手机号码、密码是否相等,如果相等,则提示"登录成功!",存储用户登录的信息,使用 Intent 回传数据,向 Intent 中添加一组键-值对附加信息:键名为 result、键值为 ok,并将打包好的数据回传给 MainActivity,如果不相等,则提示"账号/密码错误!"。

⑧ 第 146 行至第 157 行验证用户是否已存在。

3. 运行结果

在 Eclipse 中启动模拟器,然后运行项目 OnlineCareer,当"首页"单选按钮被选中,出现用户登录页,如图 10.8 所示。



图 10.8　用户登录页

10.4.2　用户注册

用户注册为首次使用本系统的用户提供服务。

用户首次运行网上求职手机客户端系统时,当"消息"单选按钮被选中,进入用户登录页,在该页面中,单击右下方的"注册"按钮,进入用户注册页。

用户注册页由“手机号”框、“密码”框、“确认密码”框和“注册”按钮组成。

用户注册页开发步骤如下。

1. 设计布局

在 res/layout 目录下的 activity_register.xml 文件中，导入 titlebar 布局文件到本布局中，用于设置“返回”文本框、“注册”文本框。设置“手机号”文本框和编辑框、“密码”文本框和编辑框、“确认密码”文本框和编辑框、“注册”按钮。

2. 编辑代码

在 com.application.onlinecareer.activities 包下的 RegisterActivity.java 文件中，加载 activity_register.xml 布局文件，分别为控件对象绑定监听器，对输入数据进行校验和处理，验证用户是否重复注册，验证电话号码是否正确。在该文件中编辑代码如下：

```
package com.application.onlinecareer.activities;

import java.util.regex.Matcher;
import java.util.regex.Pattern;
import com.application.onlinecareer.R;
import com.application.onlinecareer.util.CommonData;
import android.support.v7.app.ActionBarActivity;
import android.text.TextUtils;
import android.annotation.SuppressLint;
import android.content.ContentValues;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;

    //定义一个类 RegisterActivity 继承 ActionBarActivity 类，且实现事件监听器接口
    @SuppressLint("NewApi")
    public class RegisterActivity extends ActionBarActivity implements
            OnClickListener {

        private TextView tv_back, tv_title;
        private EditText et_phone, et_password, et_re_password;
        private Button btn_register;

        //重写 onCreate 方法
        @Override
        protected void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
```

```java
        //调用 setContentView()方法引用 activity_register 布局
        setContentView(R.layout.activity_register);
        initView();
    }

    //定义 initView()方法
    private void initView() {
        //通过 findViewById()方法获取组件的唯一标识
        tv_title = (TextView) findViewById(R.id.tv_title);
        tv_title.setText(R.string.register);
        tv_back = (TextView) findViewById(R.id.tv_back);
        btn_register = (Button) findViewById(R.id.btn_register);

        tv_back.setOnClickListener(this);
        btn_register.setOnClickListener(this);

        et_phone = (EditText) findViewById(R.id.et_phone);
        et_password = (EditText) findViewById(R.id.et_password);
        et_re_password = (EditText) findViewById(R.id.et_re_password);

    }

    //重写 onClick()方法
    @Override
    public void onClick(View v) {
        switch (v.getId()) {
        case R.id.tv_back:
            finish();
            break;
        case R.id.btn_register:
            //调用 getText().toString()方法获取字符串信息
            String phone = et_phone.getText().toString();
            String password = et_password.getText().toString();
            String rePassword = et_re_password.getText().toString();

        if (TextUtils.isEmpty(phone)) {
                Toast.makeText(this, "手机号码不能为空!",
                        Toast.LENGTH_SHORT).show();
                return;
            }
            if (TextUtils.isEmpty(password)) {
                Toast.makeText(this, "密码不能为空!", Toast.LENGTH_SHORT).show();
                return;
            }
```

```
                if (TextUtils.isEmpty(rePassword)) {
                    Toast.makeText(this, "请再次输入密码!", Toast.LENGTH_SHORT).show();
                    return;
                }

                if(isExist(CommonData.db,phone)){
                    return;
                }

                if (isMobileNumber(phone)) {
                    if (password.equals(rePassword)) {

                        //Sqlite 处理数据
                        ContentValues cValue = new ContentValues();
                        cValue.put("tel", phone);
                        cValue.put("pwd", password);
                        cValue.put("name", "匿名");
                        cValue.put("sex", "女");
                        cValue.put("age", "21");
                        cValue.put("school", "本科");
                        cValue.put("major", "成都 XX 科技有限公司");
                        cValue.put("phone", "010-22331100");
                        //调用 insert()方法插入数据
                        CommonData.db.insert("user_table", null, cValue);

                        finish();
                        Toast.makeText(this, "注册成功!", Toast.LENGTH_SHORT).show();
                    } else {
                        Toast.makeText(this, "您两次输入密码不一致!",
                                        Toast.LENGTH_SHORT)
                                .show();
                    }
                } else {
                    Toast.makeText(this, "您输入的手机号码不对!",
                            Toast.LENGTH_SHORT).show();
                }
                break;
            }
        }

        //验证用户是否重复注册
        private boolean isExist(SQLiteDatabase db,String tel) {
            String sql = "select * from user_table where tel = '" + tel + "'";
            Cursor cursor = db.rawQuery(sql, new String[0]);
            if (cursor.getCount() != 0) {
```

```
124                 Toast.makeText(RegisterActivity.this, "用户已经存在,不能重复注册!",
125                     Toast.LENGTH_SHORT).show();
126                 cursor.close();
127                 return true;
128             }else{
129                 cursor.close();
130                 return false;
131             }
132         }
133 
134         //验证电话号码是否正确
135         public boolean isMobileNumber(String mobiles) {
136             Pattern p = Pattern
137                     .compile("^((13[0-9])|(14[5,7])|(15[^4,\\D])|(18[0-9])|(17[0,
                    6,7,8]))\\d{8}$");
138             Matcher m = p.matcher(mobiles);
139             return m.matches();
140         }
141     }
```

① 第 22 行至第 141 行定义一个类 RegisterActivity 继承 ActionBarActivity 类，且实现事件监听器接口。

② 第 31 行至第 38 行重写 onCreate()方法，其中，第 36 行调用 setContentView()方法引用 activity_register 布局，第 37 行调用 initView()方法。

③ 第 41 行至第 55 行定义 initView()方法，其中，第 43 行、第 44 行至第 46 行和第 51 行至第 53 行分别通过 findViewById()方法获取控件的唯一标识，第 48 行至第 49 行分别为控件对象设置监听器。

④ 第 61 行至第 63 行，当单击“返回”文本框时，返回登录页。

⑤ 第 64 行至第 117 行，当单击“注册”文本框时，第 66 行至第 68 行使用 getText().toString()方法分别为字符串对象 phone、password 和 rePassword 获取用户输入的手机号码、密码和再次输入密码信息；第 70 行至第 82 行分别判断字符串对象 phone、password 和 rePassword 是否为空，如果为空，则分别提示“手机号码不能为空！”“密码不能为空！”“请再次输入密码！”；第 84 行至第 86 行判断用户数据库中 CommonData.db 是否存在字符串对象 phone 的值，如果存在则返回；第 88 行至第 115 行，如果确定是移动电话号码且两次输入密码一致，则调用 insert()方法插入数据，并显示“注册成功！”，否则显示“您两次输入密码不一致！”“您输入的手机号码不对！”。

⑥ 第 120 行至第 132 行验证用户是否重复注册，如果重复注册，则显示“用户已经存在，不能重复注册！”。

⑦ 第 135 行至第 140 行验证电话号码是否正确。

3. 运行结果

在 Eclipse 中启动模拟器，然后运行项目 OnlineCareer，当“首页”单选按钮被选中，出现

用户注册页，如图 10.9 所示。

图 10.9　用户注册页

10.5　职 位 详 情

职位详情提供每一职位的详细情况。

启动网上求职手机客户端系统，当“首页”单选按钮被选中，进入首页，选择“兼职”文本框或“全职”文本框，单击某一职位条目，即进入该职位的职位详情页。

职位详情页由“职位详情”标题条，“职位名称”框、“兼职(或全职)”框、“公司名称”框、“工资”框，“人数”框、“时间”框、“地址”框、“结算”框，“我要报名”框组成。

职位详情页开发步骤如下。

1. 设计布局

略。

2. 编辑代码

在 com.application.onlinecareer.activities 包下的 JobDetailActivity.java 文件中，加载 activity_job_detail xml 布局文件，分别为控件对象绑定监听器，定义 hashMap 对象，在获取有关控件的唯一标识后，通过 hashMap.get()方法返回指定键的值。在该文件中编辑代码如下：

```
package com.application.onlinecareer.activities;

import java.io.Serializable;
import java.util.HashMap;
import com.application.onlinecareer.util.CommonData;
```

```java
import com.application.onlinecareer.R;
import android.support.v7.app.ActionBarActivity;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.ImageView;
import android.widget.TextView;
import android.widget.Toast;

    //定义一个类 JobDetailActivity 继承 ActionBarActivity 类,且实现事件监听器接口
    public class JobDetailActivity extends ActionBarActivity {

        private TextView tv_back, tv_title;
        private ImageView iv_job_icon;
        private TextView tv_job_name,tv_job_type, tv_job_company,tv_job_salary,
        tv_people,tv_time,tv_address,tv_payWay,tv_apply;
        private HashMap<String, Object> hashMap;

        //重写 onCreate()方法
        @Override
        protected void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            //调用 setContentView()方法引用 activity_job_detail 布局
            setContentView(R.layout.activity_job_detail);

            getData();
            initView();

            //为"返回"文本框绑定监听器
            tv_back.setOnClickListener(new OnClickListener() {
                @Override
                public void onClick(View arg0) {
                    finish();
                }
            });

            //为"我要报名"文本框绑定监听器,验证是否登录
            tv_apply.setOnClickListener(new OnClickListener() {
                @Override
                public void onClick(View arg0) {
                    if(!CommonData.isLogin){
                        Toast.makeText(JobDetailActivity.this, "请先登录!",
                            Toast.LENGTH_SHORT).show();
                        return;
                    }else{
```

```
                }
            }
        });
    }

    private void getData() {
        Bundle bundle = getIntent().getExtras();
        Serializable data = bundle.getSerializable("job");

        if (data != null) {
        //定义 hashMap 对象
            hashMap = (HashMap<String, Object>)data;
        } else {
            return;
        }
    }

    private void initView() {

        //通过 findViewById()方法获取控件的唯一标识
        tv_title = (TextView) findViewById(R.id.tv_title);
        tv_title.setText(R.string.job_detail);
        tv_back = (TextView) findViewById(R.id.tv_back);
        iv_job_icon = (ImageView) findViewById(R.id.iv_job_icon);
        tv_job_name = (TextView) findViewById(R.id.tv_job_name);
        tv_job_type = (TextView) findViewById(R.id.tv_job_type);
        tv_job_company = (TextView) findViewById(R.id.tv_job_company);
        tv_job_salary = (TextView) findViewById(R.id.tv_job_salary);
        tv_people = (TextView) findViewById(R.id.tv_people);
        tv_time = (TextView) findViewById(R.id.tv_time);
        tv_address = (TextView) findViewById(R.id.tv_address);
        tv_payWay = (TextView) findViewById(R.id.tv_payWay);
        tv_apply = (TextView) findViewById(R.id.tv_apply);

        //通过 hashMap.get()方法返回指定键的值
        iv_job_icon.setBackgroundResource((Integer) hashMap.get("img"));
        tv_job_name.setText((CharSequence) hashMap.get("name"));
        tv_job_type.setText((CharSequence) hashMap.get("jobType"));
        tv_job_company.setText((CharSequence) hashMap.get("company"));
        tv_job_salary.setText((CharSequence) hashMap.get("salary"));
        tv_people.setText((CharSequence) hashMap.get("people"));
        tv_time.setText((CharSequence) hashMap.get("time"));
        tv_address.setText((CharSequence) hashMap.get("address"));
        tv_payWay.setText((CharSequence) hashMap.get("payWay"));

```

```
96          }
97     }
```

① 第 16 行至第 97 行定义一个类 JobDetailActivity 继承 ActionBarActivity 类，且实现事件监听器接口。

② 第 25 行至第 54 行重写 onCreate()方法，其中，第 29 行调用 setContentView()方法引用 activity_job_detail 布局，第 31 行调用 getData()方法，第 32 行调用 initView()方法，第 35 行至第 40 行为“返回”文本框绑定监听器，第 43 行至第 53 行为“我要报名”文本框绑定监听器，验证是否登录。

③ 第 56 行至第 66 行定义 getData()方法，第 62 行定义 hashMap 对象。

④ 第 68 行至第 96 行定义 initView()方法，其中，第 71 行、第 73 行至第 83 行分别通过 findViewById()方法获取控件的唯一标识，第 86 行至第 94 行分别通过 hashMap.get()方法返回指定键的值。

3. 运行结果

在 Eclipse 中启动模拟器，然后运行项目 OnlineCareer，当“首页”单选按钮被选中，出现网上求职系统首页，选择“兼职”文本框，出现如图 10.10 所示页面，在该页面中选择“话务员”条目，出现如图 10.11 所示页面。




图 10.10　首页-兼职



图 10.11　首页-兼职-话务员

如果选择“全职”文本框，出现如图 10.12 所示页面，在该页面中选择“大堂经理”条目，出现如图 10.13 所示页面。




图 10.12　首页-全职



图 10.13　首页-全职-大堂经理

10.6　我的信息

在我的信息中，介绍个人简历和编辑资料。

10.6.1　个人简历

个人简历系用户为求职提供的个人信息。

启动网上求职手机客户端系统，当“我的”单选按钮被选中，进入我的页，如果用户已登录，单击“个人简历”文本框，即进入个人简历页。

个人简历页由“个人简历”标题条，“编辑资料”文本框、“用户名”文本框、“性别”文本框、“年龄”文本框、“学历”文本框、“工作经历”文本框、“电话号码”文本框组成。

个人简历页开发步骤如下。

1. 设计布局

在 res/layout 目录下的 activity_person. xml 文件中，导入 titlebar 布局文件到本布局中，用于设置“返回”文本框、“个人简历”文本框和“编辑资料”文本框。设置“用户名”和“用户名的值”文本框、“性别”和“性别的值”文本框、“年龄”和“年龄的值”文本框、“学历”和“学历的值”文本框、“工作经历”和“工作经历的值”文本框、“电话号码”和“电话号码的值”文本框。

2. 编辑代码

在 com. application. onlinecareer. activities 包下的 PersonActivity. java 文件中，加载 activity_person. xml 布局文件，获取控件的唯一标识，分别为控件对象绑定监听器，如果已登录，获取有关键-值对中的值。在该文件中编辑代码如下：

```
package com.application.onlinecareer.activities;

import com.application.onlinecareer.R;
import com.application.onlinecareer.util.CommonData;
import android.support.v7.app.ActionBarActivity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.TextView;

    //定义一个类 PersonActivity 继承 ActionBarActivity 类,且实现事件监听器接口
    public class PersonActivity extends ActionBarActivity implements OnClickListener{
        private TextView tv_back, tv_title,tv_right;
        private TextView tv_name,tv_sex,tv_age,tv_school,tv_major,tv_phone;

        //重写 onCreate()方法
        @Override
        protected void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);

            //调用 setContentView()方法引用 activity_person 布局
            setContentView(R.layout.activity_person);
            initView();
        }

        //重写 onResume()方法
        @Override
        protected void onResume() {
            super.onResume();
            if(CommonData.isLogin){

                //通过 get().toString()方法获取键 - 值对中的值
                tv_name.setText(CommonData.user_hashMap.get("name").toString());
                tv_sex.setText(CommonData.user_hashMap.get("sex").toString());
                tv_age.setText(CommonData.user_hashMap.get("age").toString());
                tv_school.setText(CommonData.user_hashMap.get("school").toString());
                tv_major.setText(CommonData.user_hashMap.get("major").toString());
                tv_phone.setText(CommonData.user_hashMap.get("phone").toString());
            }
```

```
41          }
42          //定义 initView()方法
43          private void initView() {
44
45              //通过 findViewById()方法获取控件的唯一标识
46              tv_title = (TextView) findViewById(R.id.tv_title);
47              tv_back = (TextView) findViewById(R.id.tv_back);
48              tv_right = (TextView) findViewById(R.id.tv_right);
49              tv_title.setText(R.string.person_detail);
50              tv_right.setVisibility(View.VISIBLE);
51              tv_right.setText("编辑资料");
52
53              tv_back.setOnClickListener(this);
54              tv_right.setOnClickListener(this);
55
56              tv_name = (TextView) findViewById(R.id.tv_name);
57              tv_sex = (TextView) findViewById(R.id.tv_sex);
58              tv_age = (TextView) findViewById(R.id.tv_age);
59              tv_school = (TextView) findViewById(R.id.tv_school);
60              tv_major = (TextView) findViewById(R.id.tv_major);
61              tv_phone = (TextView) findViewById(R.id.tv_phone);
62          }
63
64          //重写 onClick()方法
65          @Override
66          public void onClick(View v) {
67              switch (v.getId()) {
68              case R.id.tv_back:
69                  finish();
70                  break;
71              case R.id.tv_right:
72                  Intent intent = new Intent(PersonActivity.this,EditDataActivity.class);
73                  startActivity(intent);
74                  break;
75              }
76          }
77      }
```

① 第 13 行至第 77 行定义一个类 PersonActivity 继承 ActionBarActivity 类，且实现事件监听器接口。

② 第 19 行至第 25 行重写 onCreate()方法，其中，第 22 行调用 setContentView()方法引用 activity_person 布局，第 24 行调用 initView()方法。

③ 第 28 行至第 41 行重写 onResume()方法，其中，第 31 行至第 40 行，如果已登录，分别通过 get(). toString()方法获取键-值对中的值。

④ 第 28 行至第 41 行定义 initView()方法,其中,第 46 行至第 48 行和第 56 行至第 61 行分别通过 findViewById()方法获取控件的唯一标识,第 53 行至第 54 行分别为控件对象设置监听器。

⑤ 第 54 行至第 143 行重写 onClick()方法,其中,第 57 行至第 65 行,当单击“返回”文本框时,返回我的页;第 71 行至第 74 行,当单击“编辑资料”文本框时,使用 Intent 显式启动 EditDataActivity,即启动编辑资料页。

3. 运行结果

在 Eclipse 中启动模拟器,然后运行项目 OnlineCareer,当“我的”单选按钮被选中,出现网上求职系统我的页,选择“我的简历”文本框,出现个人简历页,如图 10.14 所示。



图 10.14　个人简历

10.6.2　编辑资料

编辑资料为用户提供编辑个人简历的窗口。

启动网上求职手机客户端系统,当“我的”单选按钮被选中,进入我的页,如果用户已登录,单击“个人简历”文本框,进入个人简历页,其中,单击“编辑资料”文本框,进入编辑资料页。

编辑资料页由“编辑资料”标题条,“保存”文本框,“用户名”文本框和编辑框、“性别”文本框和“男”单选按钮及“女”单选按钮、“年龄”文本框和编辑框、“学历”文本框和编辑框、“工作经历”文本框和编辑框、“电话号码”文本框和编辑框组成。

编辑资料页开发步骤如下。

1. 设计布局

在 res/layout 目录下的 activity_edit_data. xml 文件中，导入 titlebar 布局文件到本布局中，用于设置“返回”文本框、“编辑资料”文本框和“保存”文本框。设置“用户名”和“用户名的值”文本框、“性别”文本框、“男”和“女”单选按钮、“年龄”和“年龄的值”文本框、“学历”和“学历的值”文本框、“工作经历”和“工作经历的值”文本框、“电话号码”和“电话号码的值”文本框。

2. 编辑代码

在 com. application. onlinecareer. activities 包下的 EditDataActivity. java 文件中，加载 activity_edit_data. xml 布局文件，分别为控件对象绑定监听器，获取用户在编辑框和单选按钮中输入的信息，将获取的用户信息向 hashMap 对象添加键-值对，保存数据。在该文件中编辑代码如下：

```
package com.application.onlinecareer.activities;

import java.util.HashMap;
import com.application.onlinecareer.R;
import com.application.onlinecareer.util.CommonData;
import android.support.v7.app.ActionBarActivity;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.EditText
import android.widget.RadioButton;
import android.widget.TextView;
import android.widget.Toast;

//定义一个类 EditDataActivity 继承 ActionBarActivity 类，且实现事件监听器接口
public class EditDataActivity extends ActionBarActivity implements OnClickListener {
        private TextView tv_back, tv_title, tv_right;
        private EditText et_name, et_age, et_school, et_major,et_phone;
        private RadioButton rb_man, rb_woman;

        //重写 onCreate()方法
        @Override
        protected void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            //调用 setContentView()方法引用 activity_edit_data 布局
            setContentView(R.layout.activity_edit_data);
            initView();

            if (CommonData.isLogin) {

                //通过 get().toString()方法获取键 - 值对中的值
```

```
                et_name.setText(CommonData.user_hashMap.get("name").toString());
                et_name.setSelection(CommonData.user_hashMap.get("name").toString()
                        .length());
                et_age.setText(CommonData.user_hashMap.get("age").toString());
                et_school.setText(CommonData.user_hashMap.get("school").toString());
                et_major.setText(CommonData.user_hashMap.get("major").toString());
                et_phone.setText(CommonData.user_hashMap.get("phone").toString());
                if (CommonData.user_hashMap.get("sex").toString().equals("男")) {
                    rb_man.setChecked(true);
                } else {
                    rb_woman.setChecked(true);
                }
            }
        }
        //定义 initView()方法
        private void initView() {

            //通过 findViewById()方法获取控件的唯一标识
            tv_title = (TextView) findViewById(R.id.tv_title);
            tv_title.setText("编辑资料");

            tv_back = (TextView) findViewById(R.id.tv_back);
            tv_right = (TextView) findViewById(R.id.tv_right);
            tv_right.setVisibility(View.VISIBLE);
            tv_right.setText("保存");

            tv_back.setOnClickListener(this);
            tv_right.setOnClickListener(this);

            et_name = (EditText) findViewById(R.id.et_name);
            et_age = (EditText) findViewById(R.id.et_age);
            et_school = (EditText) findViewById(R.id.et_school);
            et_major = (EditText) findViewById(R.id.et_major);
            et_phone = (EditText) findViewById(R.id.et_phone);
            rb_man = (RadioButton) findViewById(R.id.rb_man);
            rb_woman = (RadioButton) findViewById(R.id.rb_woman);
        }

        //重写 onClick()方法
        @Override
        public void onClick(View v) {
            switch (v.getId()) {
            case R.id.tv_back:
                finish();
                break;
```

```
                case R.id.tv_right:
                    //保存更改的用户信息
                    saveUserInfo();
                    break;
                }
            }
            //定义 saveUserInfo()方法
            private void saveUserInfo() {
                //获取用户的相关信息
                String name = et_name.getText().toString();
                String age = et_age.getText().toString();
                String school = et_school.getText().toString();
                String major = et_major.getText().toString();
                String phone = et_phone.getText().toString();
                String sex;
                if (rb_man.isChecked()) {
                    sex = "男";
                } else {
                    sex = "女";
                }

                //获取登录用户数据
                String tel = CommonData.user_hashMap.get("tel").toString();
                String pwd = CommonData.user_hashMap.get("pwd").toString();

                HashMap<String, Object> hashMap = new HashMap<String, Object>();
                hashMap.put("tel", tel);
                hashMap.put("pwd", pwd);
                hashMap.put("name", name);
                hashMap.put("sex", sex);
                hashMap.put("age", age);
                hashMap.put("school", school);
                hashMap.put("major", major);
                hashMap.put("phone", phone);

                //修改 SQL 语句
               String sql = "update user_table set name = '" + name + "', sex = '" + sex + "', age = '
               " + age + "',
                   school = '" + school + "', major = '" + major + "', phone = '" + phone + "' where tel =
                   '" + tel + "'";
               //执行 SQL 语句
               CommonData.db.execSQL(sql);

                //存在本地
                CommonData.user_hashMap = hashMap;
```

```
120                 Toast.makeText(this, "保存成功", Toast.LENGTH_SHORT).show();
121                 finish();
122             }
123 
124             //返回保存用户在 list 中的 item
125             public int userIndex(String tel) {
126                 for (int i = 0; i < CommonData.list.size(); i++) {
127                     if (CommonData.list.get(i).get("tel").equals(tel)) {
128                         return i;
129                     }
130                 }
131                 return -1;
132             }
133     }
```

① 第 16 行至第 133 行定义一个类 EditDataActivity 继承 ActionBarActivity 类，且实现事件监听器接口。

② 第 23 行至第 45 行重写 onCreate()方法，其中，第 26 行调用 setContentView()方法引用 activity_edit_data 布局，第 27 行调用 initView()方法，第 29 行至第 44 行，如果已登录，分别通过 get().toString()方法获取键-值对中的值。

③ 第 47 行至第 68 行定义 initView()方法，其中，第 50 行、第 53 行至第 54 行和第 61 行至第 67 行分别通过 findViewById()方法获取控件的唯一标识，第 58 行至第 59 行分别为控件对象设置监听器。

④ 第 71 行至第 82 行重写 onClick()方法，其中，第 74 行至第 76 行，当单击“返回”文本框时，返回个人简历页；第 77 行至第 80 行，当单击“保存”文本框时，保存更改的用户信息。

⑤ 第 84 行至第 122 行定义 saveUserInfo()方法，第 86 行至第 96 行获取用户在编辑框和单选按钮中输入的信息，第 99 行至第 100 行获取登录用户数据，第 102 行至第 110 行将获取的用户信息向 hashMap 对象添加键-值对，第 113 行至第 116 行修改 SQL 语句并执行，第 119 行至第 121 行保存数据在 user_hashMap 中并显示“保存成功”。

3. 运行结果

在 Eclipse 中启动模拟器，然后运行项目 OnlineCareer，当“我的”单选按钮被选中，出现网上求职系统我的页，选择“我的简历”文本框，出现个人简历页，选择“编辑资料”文本框，出现编辑资料页，如图 10.15 所示。



图 10.15　编辑资料

10.7 小　　结

本章主要介绍了以下内容：

(1) 在网上求职手机客户端系统需求分析和设计中，介绍了需求分析、系统目标、技术特点、系统模块结构图和表结构设计。

(2) 为开发网上求职手机客户端系统，在 Eclipse 中创建了应用项目 OnlineCareer，其程序结构包含以下 4 类文件：Activity 类和 Fragment 类文件，Adapter 类和公共数据类文件，布局文件和其他资源文件。

(3) 为了实现灵活、动态的界面设计，网上求职手机客户端系统设计采用 Fragment 划分用户界面，将 Activity 提供的用户界面划分为上下两个部分，在 Activity 运行时，Fragment 将嵌入到用户界面的上部。

基本页面是用户界面中的重要部分，基本页面由首页、消息页、我的页组成。

(4) 启动网上求职手机客户端系统，即进入首页，或当“首页”单选按钮被选中，也进入首页，此时，OneFragment 将嵌入到首页上部。

首页由首页上部和首页下部组成。其中，首页下部由“首页”单选按钮、“消息”单选按钮、“我的”单选按钮组成，其组成文件为 MainActivity.java 和 activity_main.xml；首页上部由“兼职”“全职”“职位名称”“公司名称”，“时间”“工资”等栏目组成，其组成文件为 OneFragment.java 和 fragment_one.xml。

(5) 启动网上求职手机客户端系统，若用户已登录，当“消息”单选按钮被选中，进入消息页，此时 TwoFragment 将嵌入到消息页上部；若用户未登录，当“消息”单选按钮被选中，进入用户登录页。

消息页由消息页上部和消息下部组成。其中，消息页下部由“首页”单选按钮、“消息”单选按钮、“我的”单选按钮组成，其组成文件为 MainActivity.java 和 activity_main.xml；消息页上部由“消息”“已录用”“待录用”“已报名”等栏目组成，其组成文件为 TwoFragment.java 和 fragment_two.xml。

(6) 启动网上求职手机客户端系统，当“我的”单选按钮被选中，进入我的页，此时 ThreeFragment 将嵌入到我的页上部。

我的页由我的页上部和我的页下部组成。其中，我的页下部由“首页”单选按钮、“消息”单选按钮、“我的”单选按钮组成，其组成文件为 MainActivity.java 和 activity_main.xml；我的页上部由“我的”“我的简历”“关于产品”“撤销登录”等栏目组成，其组成文件为 ThreeFragment.java 和 fragment_three.xml。

(7) 启动网上求职手机客户端系统后，如果用户尚未登录，当“消息”单选按钮被选中，即进入用户登录页，如果用户是首次登录单击右下方的“注册”按钮，进入用户注册页。

用户登录页由“返回”文本框、“手机号”编辑框、“密码”编辑框、“登录”按钮和“注册”文本框组成，其组成文件为 LoginActivity.java 和 activity_login.xml。

用户注册页由“手机号”框、“密码”框、“确认密码”框和“注册”按钮组成，其组成文件为 RegisterActivity.java 和 activity_register.xml。

(8) 启动网上求职手机客户端系统，当“首页”单选按钮被选中，进入首页，选择“兼职”文本框或“全职”文本框，单击某一职位条目，即进入该职位的职位详情页。

职位详情页由“职位详情”标题条，“职位名称”框、“兼职(或全职)”框、“公司名称”框、“工资”框，“人数”框、“时间”框、“地址”框、“结算”框，“我要报名”框组成，其组成文件为JobDetailActivity.java 和 activity_job_detail.xml。

(9) 启动网上求职手机客户端系统，当“我的”单选按钮被选中，进入我的页，如果用户已登录，单击“个人简历”文本框，即进入个人简历页，单击“编辑资料”文本框，进入编辑资料页。

个人简历页由“个人简历”标题条，“编辑资料”文本框、“用户名”文本框、“性别”文本框、“年龄”文本框、“学历”文本框、“工作经历”文本框、“电话号码”文本框组成，其组成文件为PersonActivity.java 和 activity_person.xml。

编辑资料页由“编辑资料”标题条，“保存”文本框，“用户名”文本框和编辑框、“性别”文本框和“男”单选按钮及“女”单选按钮、“年龄”文本框和编辑框、“学历”文本框和编辑框、“工作经历”文本框和编辑框、“电话号码”文本框和编辑框组成，其组成文件为EditDataActivity.java 和 activity_edit_data.xml。

习 题 10

一、选择题

10.1 应用项目 OnlineCareer 的基本页面不包括________。

A. 消息页　　B. 用户登录页　　C. 首页　　D. 我的页

10.2 Fragment 类的文件不包括________。

A. ThreeFragment.java　　B. OneFragment.java

C. CommonData.java　　D. TwoFragment.java

二、填空题

10.3 用户注册并登录后，才能运行该系统的全部功能，未注册的用户仅能________。

10.4 已注册的用户未进行登录，仅能浏览部分页面，已注册的用户登录后，才能运行该系统的________。

10.5 MainActivity.java 用于显示 activity_main 布局文件提供的________，创建数据库和表，为其中的控件设置监听器资源等。

10.6 fragment_one 布局文件，包含________、“首页”文本框、“兼职”文本框、“全职”文本框等。

10.7 titlebar 布局文件，包含“________”文本框、“标题”文本框、“进入”文本框等。

10.8 网上求职手机客户端系统设计采用 Fragment 划分用户界面，将 Activity 提供的用户界面划分为________。

10.9 应用项目 OnlineCareer 启动后，Fragment 将嵌入到________。

10.10 启动网上求职手机客户端系统，若用户未登录，当“消息”单选按钮被选中，进入________。

三、问答题

10.11　简述网上求职手机客户端系统的系统目标和技术特点。

10.12　网上求职手机客户端系统的程序结构包含哪几类文件?

10.13　分别简述首页、消息页、我的页的组成。

10.14　Fragment 的生命周期受到所在的 Activity 哪些影响?

10.15　实现 listView 控件,有哪几个步骤?

10.16　分析采用 Intent 传递数据以区分已登录状态和未登录状态的关键代码。

10.17　分析创建 SQLite 数据库和表的关键代码。

10.18　分析 CommonData 类保存公共数据的有关代码。

四、应用题

10.19　将用 SQLite 数据库存储数据的代码改为用 SharedPreferences 存储数据。

10.20　完善网上求职手机客户端系统的功能,例如继续开发已录用页、待录用页、已报名页。

习题参考答案

第 1 章　Android 系统体系架构和应用开发环境

一、选择题

1.1　D　　1.2　B

二、填空题

1.3　强大的应用开发平台

1.4　商务

1.5　Wi-Fi 驱动

1.6　SDK

1.7　ADT

1.8　IntelliJ IDEA

三、问答题

略

四、应用题

略

第 2 章　Android 应用的创建、调试和发布

一、选择题

2.1　C　　2.2　B　　2.3　D　　2.4　A　　2.5　B　2.6　D　2.7　C　2.8　D

2.9　A　　2.10　C

二、填空题

2.11　字符串

2.12　XML

2.13　Java

2.14　布局文件

2.15　LogCat

2.16　错误信息

2.17　编译、打包、安装、运行

三、问答题

略

四、应用题

略

第 3 章　Activity、Fragment 和 Intent

一、选择题

3.1　D　　3.2　B　　3.3　A　　3.4　D　　3.5　D　　3.6　C　　3.7　C
3.8　B　　3.9　B　　3.10　A

二、填空题

3.11　Intent
3.12　运行状态
3.13　onStart()
3.14　Category
3.15　Data

三、问答题

略

四、应用题

略

第 4 章　Android 基本控件

一、选择题

4.1　B　　4.2　C　　4.3　A　　4.4　D　　4.5　B　　4.6　A　　4.7　A
4.8　C

二、填空题

4.9　信息交换
4.10　可视化窗体控件
4.11　相对布局
4.12　多个
4.13　相对位置
4.14　android.widget
4.15　编辑功能
4.16　图片
4.17　图片对象
4.18　秒

三、问答题

略

四、应用题

略

第 5 章 Android 事件处理、高级控件和菜单

一、选择题

5.1 B 5.2 C 5.3 C 5.4 B 5.5 D 5.6 A 5.7 D
5.8 C

二、填空题

5.9 Event Listener
5.10 垂直下拉
5.11 垂直
5.12 切换
5.13 Gallery
5.14 ProgressBar
5.15 Menu
5.16 相似功能
5.17 OptionMenu

三、问答题

略

四、应用题

略

第 6 章 后台服务

一、选择题

6.1 C 6.2 D 6.3 B 6.4 A 6.5 C 6.6 B

二、填空题

6.7 onCreate()
6.8 BroadcastReceiver
6.9 bindService
6.10 隐式启动
6.11 unbindService()
6.12 其他组件
6.13 Notification

三、问答题

略

四、应用题

略

第 7 章 数据存储

一、选择题

7.1 B 7.2 D 7.3 C 7.4 B 7.5 C

二、填空题

7.6　键-值对

7.7　轻量级

7.8　帮助类

7.9　创建

7.10　更新

7.11　共享数据

7.12　可用资源

7.13　URI

三、问答题

略

四、应用题

略

第 8 章　多媒体服务

一、选择题

8.1　B　　8.2　D　　8.3　C

二、填空题

8.4　Graphics

8.5　Paint

8.6　三角形

8.7　一系列

8.8　旋转

8.9　MediaPlayer

8.10　VideoView

8.11　MediaRecorder

三、问答题

略

四、应用题

略

第 9 章　定位服务和百度地图应用开发

一、选择题

9.1　C　　9.2　D

二、填空题

9.3　基于位置的服务

9.4　定位

9.5　电子商务模式

9.6　地理位置服务
9.7　第三方提供的地图应用接口
9.8　接口
9.9　申请密钥
9.10　应用开发项目

三、问答题

略

四、应用题

略

第 10 章　Android 应用项目开发

一、选择题

10.1　B　　10.2　C

二、填空题

10.3　浏览部分页面
10.4　全部功能
10.5　用户界面
10.6　列表视图
10.7　返回
10.8　上下两个部分
10.9　用户界面的上部
10.10　用户登录页

三、问答题

略

四、应用题

略

参 考 文 献

[1] ANNOZZI J,Jr,DARCEV L,CONDER S. Android应用程序开发权威指南.4版.林学森,周昊来,译.北京：电子工业出版社，2015.

[2] DEITEL P,DEITEL H,DEITEL A. Android大学教程.2版.胡彦平,张君施,闫锋欣,译.北京:电子工业出版社，2015.

[3] 王向辉,张国印,沈洁. Android应用程序开发.3版.北京：清华大学出版社，2016.

[4] 张冬玲,杨宁. Android应用开发教程.北京：清华大学出版社，2013.

[5] 李刚.疯狂Android讲义.2版.北京：电子工业出版社，2013.

[6] 董志鹏,张水波. Android手机应用开发简明教程.北京：清华大学出版社，2016.

图 书 资 源 支 持

感谢您一直以来对清华版图书的支持和爱护。为了配合本书的使用，本书提供配套的资源，有需求的读者请扫描下方的“书圈”微信公众号二维码，在图书专区下载，也可以拨打电话或发送电子邮件咨询。

如果您在使用本书的过程中遇到了什么问题，或者有相关图书出版计划，也请您发邮件告诉我们，以便我们更好地为您服务。